SIMULATING HUMAN MOTION FOR OBJECT MANIPULATION

A Thesis
Presented to
The Academic Faculty

by

Yunfei Bai

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology
December 2015

SIMULATING HUMAN MOTION FOR OBJECT MANIPULATION

Approved by:

Professor C.Karen Liu, Advisor
School of Interactive Computing
Georgia Institute of Technology

Professor Andrea Thomaz
School of Interactive Computing
Georgia Institute of Technology

Professor Greg Turk
School of Interactive Computing
Georgia Institute of Technology

Professor Paul Kry
School of Computer Science
McGill University

Professor Jarek Rossignac
School of Interactive Computing
Georgia Institute of Technology

Date Approved: October 22, 2015

ACKNOWLEDGEMENTS

First and foremost, I want to thank my advisor Professor C.Karen Liu for her guidance and support. It has been an honor to be her Ph.D. student. I appreciate all her contributions of time and idea to make my Ph.D. experience productive, and rewarding. I also want to thank my committee members Professor Greg Turk, Professor Jarek Rossignac, Professor Andrea Thomaz, and Professor Paul Kry for their advice, time, comments, and help.

The members of the computer graphics group have contributed immensely to both my professional and personal time at Georgia Tech. The group has been a source of friendship, good advice, and collaboration. Especially, I am grateful for the fun group who stuck it out in grad school with me: Jie Tan, Kristin Siu, Wenhao Yu, Sehoon Ha, Sumit Jain, Yuting Ye, Yuting Gu, Mark Luffel, Karthik Raveendran, Tina Zhuo, Jeongseok Lee, Mukul Sati and Alexander Clegg.

Also, I would not be able to survive and stay sane in grad school without a good support system: friends. Yuzhong Zhang, Yangfeng Ji, Feifei Qian, Hai Shang, Zainan Zhou, Xuan Qin, and many others made my time enjoyable and have become a part of my life. Thank you all for your inexhaustible encouragement and the part you all played in helping me get to this point. I cannot tell you how lucky I am to have your friendship.

Lastly, I would like to thank my family for all their love and encouragement. My family is always my inspiration and they have been very supportive of me all these years. My parents always believe in me and support me whatever decisions I make as a whole person and as a responsible being. And most of all, I feel extremely lucky to meet my loving, supportive, and patient fiancee Sijie at Georgia Tech. Thank you

for always being there for me. Thank you for your unconditional love. I could not ask for more.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Performing object manipulation with full-body and dexterous hands is an essential human activity in everyday environment. Although problems involving object manipulation have been frequently studied by researchers in computer animation, virtual characters exhibiting the same level of versatility as their human counterparts have not been demonstrated. In this thesis, we identify three aspects of versatile manipulation skills based on the observation of human behaviors. First, humans are adept at organizing a set of independent tasks into consecutive and concurrent tasks that maximize efficiency – putting the coffee mug on the roof of the car, opening the car door, and loading stuff into the car while holding a baby. Second, humans are good at utilizing different body parts for manipulation tasks – using the elbow to push the door when both hands are occupied or using both palm and fingers to orient a cellphone in hand. Third, humans are skilled in manipulating a variety set of objects with different material properties – the same manipulators (that is hands) are capable of folding laundry and hammering a nail.

One existing challenge is to develop algorithms that can coordinate different object manipulation tasks. Moreover, it requires solving challenging planning and control problems to capture how humans deftly exploit different properties of different body parts. In addition, manipulation of deformable objects brings challenges in physical simulation and dynamic control due to complex collision phenomena and large number of degrees of freedom for the object.

We use physics-based optimization and control techniques to simulate human motions for full-body and dexterous hand manipulation with versatility that humans exhibit, to address the abovementioned challenges. This thesis focuses on three research

problems: (1) synthesize human activities involving concurrent full-body manipulation of multiple objects; (2) synthesize in-hand manipulation of a polygonal object with integrated control techniques for both palm and fingers; and (3) synthesize realistic dexterous manipulation of cloth from a simple description of the desired cloth motion. The algorithms presented in this thesis make a step further towards automatically creating animation for full-body and dexterous hand object manipulation with human versatility.

CHAPTER I

INTRODUCTION

Performing object manipulation with full-body and dexterous hands is an essential human activity in everyday environments. A mundane morning routine before going to work can involve numerous object manipulation tasks: getting dressed, picking up and turning on the cellphone, tucking the briefcase under the arm so the hand can search for keys in the pocket, and pushing the front door open by leaning on it. These mundane tasks are easy for humans to perform but require sophisticated algorithms for virtual characters to imitate.

Inspired by human motion, researchers in the computer animation area have made tremendous progress in developing algorithms for virtual characters to imitate human motions for object manipulation. Meanwhile, the problem of object manipulation has also been studied extensively in the robotics research (Figure 1). The ideas and knowledge body have been shared frequently in these two research communities. With the developments in these areas, animation creation processes can be automatic and less time consuming. However, the motions of virtual characters that can be achieved with the current algorithms lack the realism capacity to recreate the **versatility** that humans can exhibit in object manipulation tasks. The word versatility describes a person having many different skills or qualities, which allows him or her to adapt to many different situations. In the context of object manipulation, we identify three aspects of versatile manipulation skills based on the observation of human behaviors, namely, the ability to organize a set of independent tasks into consecutive and concurrent ones, the ability to utilize different body parts, and the ability to manipulate objects with different material properties. These three aspects of human versatility

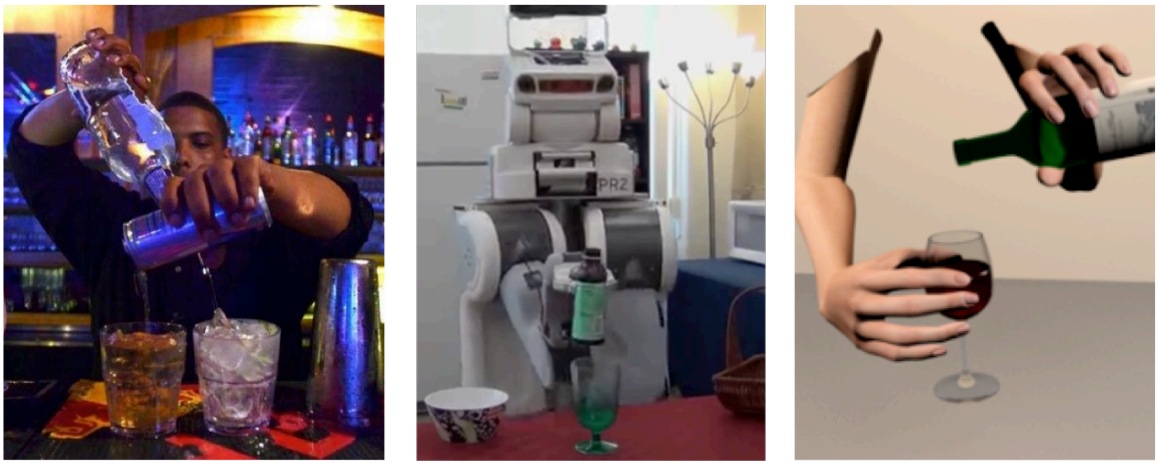


Figure 1: Robots and virtual characters imitate human motions for object manipulation. Left: Bartender from Google image. Middle: Learning sequences of controllers for complex manipulation tasks [122]. Right: Physics-based motion synthesis for creating hand manipulation [79].

bring significant challenges to develop algorithms for animated virtual characters to imitate human motions for object manipulation.

Humans are adept at organizing a set of independent tasks into concurrent tasks that maximize efficiency. For example, a person can hold a mug, carry a bag, and push the door open with his or her elbow at the same time. In computer graphics, synthesizing human multitasking has not been broadly explored in the research problem of full-body manipulation. Simulating human multitasking for object manipulation requires sophisticated control that can coordinate different object manipulation tasks without interference among them.

Humans are good at utilizing different body parts for manipulation tasks. For example, we can roll an cellphone on the palm or rotate it with fingers. Moreover, people can carry an basket using our shoulders, elbows, or even our head. The existing control algorithms for both full-body and dexterous hand manipulation only utilize designated parts of virtual characters for object manipulation. To automatically create realistic animations for object manipulation requires solving challenging planning and control problems to capture how humans deftly exploit different properties of

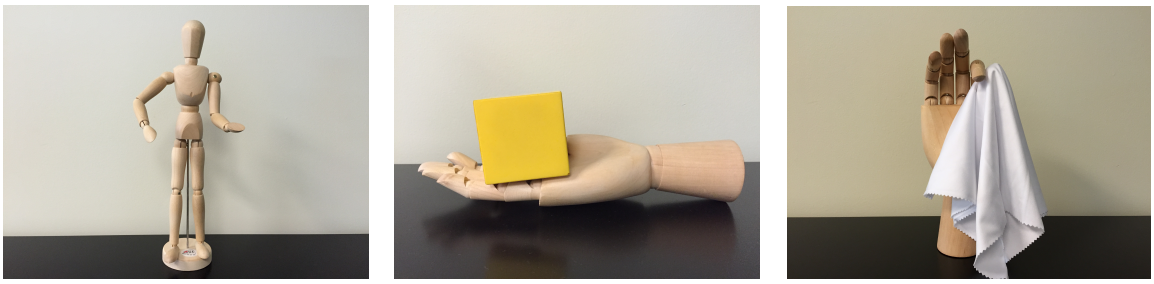


Figure 2: Three research topics of this thesis. Left: Synthesize human activities involving concurrent full-body manipulation of multiple objects. Middle: Synthesize in-hand manipulation of a polygonal object with integrated control techniques for both palm and fingers. Right: Synthesize dexterous manipulation of cloth from a simple description of the desired cloth motion.

different body parts and objects.

Humans are skilled in manipulating a variety set of objects with different material properties. Even though many algorithms in computer graphics have been proposed in recent years to create realistic animation of dexterous hand manipulation, the objects are typically rigid. This leaves a large gap in the animation space, since many manipulation tasks in daily life involve non-rigid objects, such as wringing a towel or folding a shirt. Synthesizing dexterous hand manipulation of cloth is a challenging problem in terms of both physical simulation and dynamic control due to more complex collision phenomena and more degrees of freedom of the object compared to rigid object manipulation.

This thesis aims to create virtual characters with the level of versatility that humans exhibit by developing physics-based optimization and control algorithms for full-body and dexterous hand manipulation tasks. Specifically, this thesis addresses three research topics (Figure 2):

1. Synthesize human activities involving concurrent full-body manipulation of multiple objects.

2. Synthesize in-hand manipulation of a polygonal object with integrated control techniques for both palm and fingers.

3. Synthesize dexterous manipulation of cloth from a simple description of the desired cloth motion.

1.1 Thesis Overview

This dissertation presents several algorithms for synthesizing full-body and dexterous hands performing object manipulation tasks. We start with introducing the motivation, explaining the challenges, and defining the problems we address in this thesis (Chapter 1). We then describe the existing kinematic and dynamic approaches to synthesize motions of virtual characters, especially in terms of the full-body object manipulation and the dexterous hand manipulation (Chapter 2). We present a set of new computational tools to improve the versatility in object manipulation motions which focus on: (1) human multitasking that exploits different body parts and organizes a set of tasks into concurrent ones (Chapter 3), (2) manipulating the orientation of an object using different hand parts (Chapter 4), and (3) manipulating non-rigid objects through solving challenging physical simulation problem (Chapter 5) and dynamic control problem (Chapter 6). We conclude the thesis with the discussion of the limitations, applications, and future extensions of this research (Chapter 7).

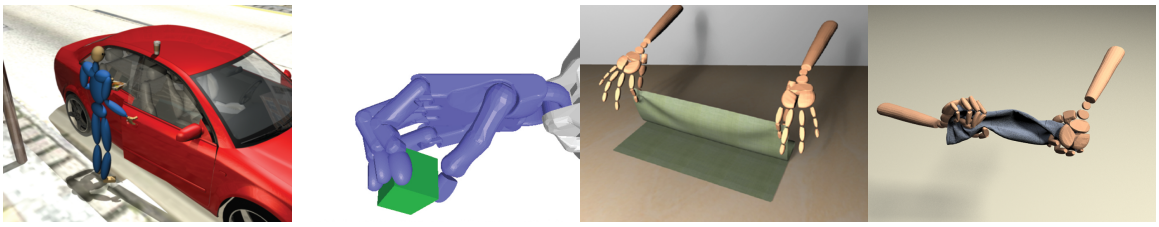


Figure 3: Representative images of the work presented in this thesis. (a) Synthesis of concurrent object manipulation tasks. (b) Dexterous manipulation using both palm and fingers. (c) Coupling cloth and rigid bodies for dexterous manipulation. (d) Dexterous manipulation of cloth.

1.1.1 Synthesis of Concurrent Object Manipulation Tasks

Performing multiple objects manipulation tasks concurrently is a common human activity in everyday environments. A mundane morning routine before going to work can involve numerous consecutive and concurrent tasks: picking up the briefcase, tucking the briefcase under the arm so the hand can search for keys in the pocket, and pushing the front door open by leaning on it. In Chapter 3, we introduce a physics-based technique to synthesize human activities involving full-body manipulation of multiple objects (Figure 3(a)). With an abstract representation of objects and environment as input, our method creates a continuous animation of a virtual character manipulating multiple objects concurrently at various locations in the environment.

Making virtual characters imitate human motions for multitasking requires solving complicated planning and dynamic motion control problems to capture how humans deftly exploit different body parts and coordinate concurrent object manipulation tasks. For solving the planning problem, we introduce a graph structure to describe how each object can be manipulated using different strategies. The problem of manipulation planning can then be transformed to a standard graph traversal. For solving the control problem, we present two new ideas in the framework of operational space control. One is a "task consistency" metric to determine when to overlap tasks and when to execute them in succession, the other is to solve a desired pose so that the character can make adjustment to improve the "task consistency" for future tasks. Compared to inverse kinematics and motion planning techniques, our method can generate more physically realistic motions, especially when many manipulation tasks are involved. Since our method dynamically simulates the character's motion, the character can react differently to objects with different physical properties.

One limitation is that we only consider the shortest traveling distance of the character when planning the tasks. One future work direction could be taking other

criteria into account for planning the tasks, such as the amount of effort required for each task.

1.1.2 Dexterous Manipulation Using Both Palm and Fingers

Besides full-body manipulation, using hands to perform dexterous manipulation tasks is an important piece to complete a fully simulated virtual character for object manipulation. While there has been much research in computer animation on dexterous hand manipulation, most of the work only utilize designated parts of hand and focus on a particular manipulation strategy such as grasping or finger gating. To improve a character's versatility by using different hand parts, in Chapter 4, we present a technique to manipulate the orientation of an object using both palm and fingers of a virtual hand (Figure 3(b)). Given a polygonal object on the table, our method generates a controlled motion of a virtual hand that grasps the object from table, transports the object to a specific spot on palm, rolls the object on palm to a desired configuration, and corrects the rolling deviation using fingers.

We develop two types of controllers for both palm and fingers through different manipulation strategies. For palm control, we solve the non-prehensile manipulation problem which manipulates an object without using fingers to grasp. To achieve this, we formulate a sequence of inverted pendulum problems to control an object rolling from an initial configuration to a desired configuration on palm. Compared to using fingers to grasp, this manipulation strategy has the flexibility of manipulating objects with various sizes and shapes. We demonstrate our palm controller on different objects with different rolling times. For finger control, we develop a multi-finger controller for stable grasping and correcting the rolling deviation of an object.

One limiting factor of our algorithm is that the wrist tilting angle needs to be bounded by the friction coefficient to prevent slippage. As for future work, one direction would be handling an object that is very different from a prism shape, such

as rolling a key.

1.1.3 Coupling Cloth and Rigid Bodies for Dexterous Manipulation

Manipulating cloth such as folding laundry is an important daily skill. Even though many algorithms in computer graphics have been proposed for dexterous hand manipulation, the objects are typically rigid. In animation, automatic solutions for dexterous manipulation of non-rigid objects have not been explored. One challenge is that there is no simulator that can accurately and efficiently simulate the complex interaction between hands and cloth, due to incorrect computation of contact forces and collision issues. In Chapter 5, we present a simulation technique that couples rigid body and cloth simulations with the existing rigid body and cloth simulators as black box (Figure 3(c)).

Our method solves two main issues in the current cloth simulators. The first issue is that the contact force computation does not take into account the dynamic information of the rigid body. Most current cloth simulators compute contact forces based only on the position and the velocity of the rigid body. The consequence of an inaccurate contact force is particularly evident when the rigid body is not interpenetrating the cloth but is imparting force on the cloth. The second issue is that the rigid body motion is unaware of the state of cloth. Therefore the cloth can be trapped by the rigid bodies from two sides, which can frequently cause unsolvable collision situations. To resolve these two issues, we develop a light-weight interface, rigid cloth patch, through which the rigid body and the cloth simulations communicate with each other. Moreover, we formulate a nonlinear complementarity problem to consider all the objects in contact with each cloth vertex simultaneously, leading to more accurate friction forces compared to the sequential computation in most cloth simulators. We apply our method to a variety of cloth manipulation scenarios, and demonstrate the benefit of our method by showing the motion that cannot be achieved with naïve

position constraints.

We make the decision not to implement accurate two-way coupling for the reason of having a simple implementation and efficient computation. However, we do recognize that some examples like putting on a sock requires that the rigid bodies react to the dynamics of the cloth accurately.

1.1.4 Dexterous Manipulation of Cloth

With a working simulator, an experienced artist might be able to animate simple cloth manipulation tasks through trial-and-error, but more complex interaction demands more sophisticated automatic solution. In Chapter 6, we introduce an approach to control the cloth motion to follow the desired motion trajectory specified by user through the actions of hands.

In computer graphics, researchers have developed various algorithms to control physically simulated cloth according to user-specifications, such as making the cloth hanging on the bar, or lifting a corner of the cloth. These techniques allow unphysical forces to be applied anywhere on the cloth without consideration of hands. Therefore, dexterous hand manipulation and user-control of cloth simulation are two active computer animation research areas that have rarely crossed path. We propose a technique to consolidate the constraints imposed by dexterous hand manipulation and control of cloth simulation. In our framework, the artist provides the desired trajectories of cloth and the desired grasp points on the cloth. We formulate a model predictive control (MPC) problem to solve the desired commanding forces from hands to cloth, such that through coupled dynamic simulation, the cloth can follow the user-specified motion trajectories closely.

One assumption of this technique is that the manipulation strategy is to maintain static static during manipulation. We demonstrate that this conservative strategy can achieve a variety of manipulation tasks in daily activities. However, more general

manipulation task such as sliding two hands across the cloth to remove wrinkles, a manipulation strategy that can utilize dynamic friction is very useful.

1.2 Contributions

The work in this thesis has several contributions to simulating natural human motions for object manipulation.

A framework for synthesizing concurrent object manipulation tasks. We introduce a physics-based technique to synthesize human activities involving concurrent full-body manipulation of multiple objects. We design dynamic controllers to physically simulate upper-body manipulation and integrate it with procedurally generated locomotion and hand grasping motion. We introduce a graph structure, a manipulation graph, to describe how each object can be manipulated using different strategies. Our control algorithm optimally schedules and executes multiple tasks based on the dynamic space of the tasks and the state of the character. We introduce a task consistency metric to measure the physical feasibility of multitasking. Furthermore, we exploit the redundancy of control space to improve the character's ability to multitask. The output is a continuous animation of the character manipulating multiple objects and environment features concurrently at various locations in a constrained environment.

A control technique using both palm and fingers for dexterous manipulation. We present a technique to manipulate the orientation of an object using both palm and fingers of a virtual robotic dexterous hand. We formulate a simple algorithm to control the tilting angle of palm based on the conservation of mechanical energy and an empirical model of energy dissipation due to collisions. Additionally, we develop a corrective controller for fingers of the virtual robotic hand to improve the robustness against unexpected collision, irregular object, and noisy vision sensing input. The computation is simple and the controller can run in realtime on a virtual

robotic hand. We demonstrate the controller on the virtual Shadow Dexterous Hand model to reorient different types of objects.

A simulation technique for coupling cloth and rigid bodies. We introduce a physics based simulation technique that couples cloth and rigid body simulation to synthesize dexterous manipulation of cloth. We develop a light-weight interface through which the rigid body and cloth simulators communicate on a demand-driven manner to resolve two issues: (1) the contact force computation does not take into account the dynamic information of the rigid body; (2) rigid body motion is unaware of the state of cloth, which frequently causes unsolvable collision situations. We demonstrate our method on a set of basic cloth manipulation skills, such as gripping, pinching, and pressing.

A control technique for dexterous manipulation of cloth. We formulate an optimization problem that solves for commanding forces from simulated hands to cloth to follow a desired cloth motion target. To balance between the effectiveness of control and computational costs, we formulate a model-predictive-control problem as a quadratic program at each time step. The solved commanding forces are then used to guide the joint torques of virtual hands. We demonstrate our technique on a set of cloth manipulation tasks in daily activities, including folding laundry, wringing a towel, and putting on a scarf.

CHAPTER II

BACKGROUND

2.1 Human Motion Synthesis

Algorithms for computer animation are commonly evaluated on simplified models with joints and rigid body primitives. To evaluate our algorithms, we design articulated full-body (Figure 4) and hand models based on simplified musculoskeletal system models of humans. Each body part is represented as a rigid body and each joint has 1-3 degrees of freedom (DOFs)[1].

The main approach of this thesis is to dynamically simulate articulated characters with control solved from a physics-based optimization to satisfy the desired motion target. The algorithms presented in this thesis combine both kinematic and dynamic approaches. In this section, we discuss some kinematic and dynamic approaches that are widely used for synthesizing motion of articulated characters.

2.1.1 Kinematic Approach

Inverse kinematics (IK) routine solves for a sequence of poses of the character which best match the input motion capture data. For each frame of the motion capture data, a pose is estimated by formulating an optimization that minimizes the distance between the marker positions in the estimated pose and the corresponding marker positions in the same frame of the input data:

$$\min_{\mathbf{q}} \sum_i \omega_i \|\mathbf{h}_i(\mathbf{q}) - \mathbf{p}_i\|_2^2 + \|\mathbf{q} - \mathbf{q}'\|_2^2, \tag{1}$$

where $\mathbf{q}$ is a pose described in generalized coordinates, $\mathbf{h}$ is the position of marker i in world coordinates given $\mathbf{q}$, $\mathbf{p}_i$ is the captured position of marker i, and ω_i is the

[1]The root of the articulated character can be a free joint with 6 DOFs.

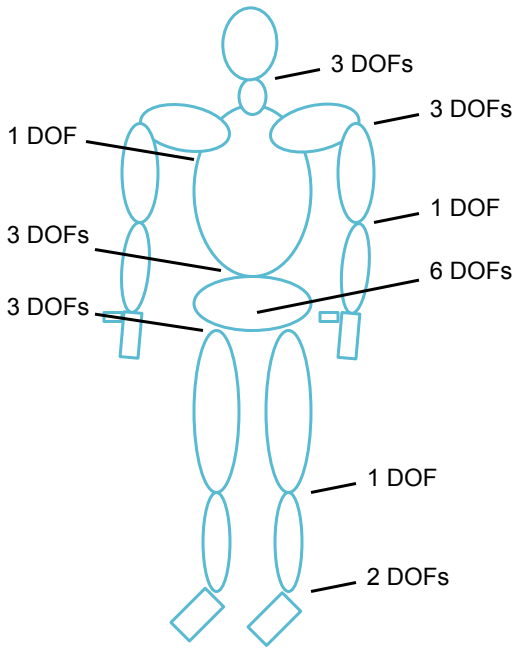


Figure 4: The articulated full-body human character used in Chapter 3 has 16 DOFs on the upper body (not including wrist and hand) and 18 DOFs on the lower body.

weight. The second term improves the smoothness between the currently estimated pose $\mathbf{q}$ and the pose solved from the previous frame $\mathbf{q}'$.

A character motion database can be created by solving IK for motion capture data. By blending existing motion data, we can create motions that are adaptive to new situations. The motion graph algorithm is a technique for constructing a directed graph using the motion database [72]. Motions can be generated by simply building walks on the graph. In Chapter 3, we create locomotion of a character by blending captured short walking sequences of straight walks, single steps, and turning motions, while maintaining correct contact states to navigate in the constrained environment, in a manner similar to the method described by Kovar et al. [72].

2.1.2 Dynamic Approach

Articulated character motion can be described by a set of dynamic equations of motion for multibody systems. Equations of motion in generalized coordinates are expressed

as follows:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{J}_p^T \mathbf{F}, \tag{2}$$

where $\mathbf{q}$ is the joint configuration, $\dot{\mathbf{q}}$ is the joint velocity, and $\ddot{\mathbf{q}}$ is the joint acceleration. In Equation 2, $\mathbf{M}$ denotes the mass matrix, $\mathbf{C}$ includes Coriolis, centrifugal, and gravitational forces, $\mathbf{J}_p$ is the Jacobian matrix that projects external force $\mathbf{F}$ applied at a body point $\mathbf{p}$ from Cartesian to generalized coordinates, and $\boldsymbol{\tau}$ represents the generalized control forces applied internally by the character. To control the character's motion, we need to compute proper $\boldsymbol{\tau}$ to satisfy the desired motion target.

Proportional-derivative (PD) control is a control loop feedback mechanism. The controller attempts to minimize the error from the target trajectory over time by adjustment of the control variable. PD control can be used at each actuated joint, acting like a spring and a damper, to provide a simple framework to compute control forces for tracking a joint trajectory. The formulation of PD control in joint space is written as follows:

$$\boldsymbol{\tau} = K_p(\bar{\mathbf{q}} - \mathbf{q}) + K_d\dot{\mathbf{q}}, \tag{3}$$

where K_p and K_d are proportional and derivative gains, and $\bar{\mathbf{q}}$ is the desired joint configuration. One common problem with PD control is that when precise tracking with high gains is desired, PD control typically produces unstable control forces which require the numerical simulation to take small time steps. Stable PD control [124] improves the stability of PD control allowing for arbitrarily high gains, even at large time steps. In Chapter 6, we use stable PD control for the grasp controller, which computes the appropriate torques to move the hand from its arbitrary initial pose to the desired grasping pose.

The problem with generalized coordinates is that planning trajectories in this space for tasks performed in Cartesian space is not straight forward. The idea behind operational space control is to control in the Cartesian coordinate system that is

directly relevant to the task that we wish to perform. We can write the relation between the acceleration of a Cartesian point $\mathbf{x}$ on the character and the joint velocity $\dot{\mathbf{q}}$ and acceleration $\ddot{\mathbf{q}}$ as follows:

$$\ddot{\mathbf{x}} = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}. \tag{4}$$

Note that $\mathbf{J}$ is the Jacobian matrix evaluated at the point $\mathbf{x}$ and is in general different from $\mathbf{J}_p$ in Equation 2. With Equation 4, we can rearrange Equation 2 to express the equations of motion in Cartesian space:

$$\mathbf{J}\mathbf{M}^{-1}\boldsymbol{\tau} = \ddot{\mathbf{x}} + \mathbf{J}\mathbf{M}^{-1}\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \dot{\mathbf{J}}\dot{\mathbf{q}} - \mathbf{J}\mathbf{M}^{-1}\mathbf{J}_p^T\mathbf{F}. \tag{5}$$

This equation represents a simple linear relation between the acceleration of a Cartesian point $\ddot{\mathbf{x}}$ for the task and the required joint torques $\boldsymbol{\tau}$. In Chapter 3, we use operational space control for the character to achieve manipulation tasks in Cartesian space. We also exploit the redundancy of the operational space control to improve the character's ability to multitask.

Besides controlling a desired pose in Cartesian space, often we wish to control the end-effector to achieve the desired Cartesian force. Jacobian transpose is a control scheme to achieve this goal [121]. The internal control torque $\boldsymbol{\tau}$ is computed based on the following relation with the desired Cartesian force $\mathbf{F}$:

$$\boldsymbol{\tau} = \mathbf{J}_p^T\mathbf{F}. \tag{6}$$

In Chapter 4 and Chapter 5, we use the Jacobian transpose control scheme to control the desired force exerted from the dexterous hand to the object being manipulated.

While PD control is widely used in controlling character's motion, it does not have the ability to anticipate future events. Model predictive control (MPC) can consider the future and take appropriate control action by utilizing the dynamic model of the system. MPC has been used to plan for control of the virtual hand for manipulation tasks by considering the future state [80]. In Chapter 6, we formulate a MPC for cloth manipulation task at each time step.

2.2 Object Manipulation Motion Synthesis

Object manipulation is a form of dexterity play or performance in which people physically interact with one or more objects. Object manipulation is an important type of human motion. It can be classified into full-body manipulation, where the entire body is used to interact with objects, and dexterous hand manipulation, where fingers and palm are used to finely control motions of objects.

2.2.1 Full-body Manipulation

Effective methods for synthesizing full-body manipulation are crucial for animating everyday human behaviors. Most previous works exploit inverse kinematics and motion planning techniques to generate motions that satisfy desired manipulation tasks. To achieve collision-free motion, these methods apply path planning algorithms in workspace [83, 135], configuration space [69, 62, 61], or configuration space with an additional timing dimension [114]. Using the result from the planning algorithm as guidance, natural-looking, full-body animation can be synthesized based on heuristics or motion capture data. In addition, the general problem of manipulating objects with locomotion has been studied previously [40, 54, 119, 30]. In Chapter 3, we focus on multitasking in full-body manipulation that exploits various manipulation strategies concurrently, which has not been broadly explored in the computer graphics domain. In addition, we take the approach of physical simulation instead of kinematic procedures or motion capture. The simulated motion exhibits greater physical realism for dynamically demanding tasks, such as holding a cup of water while lifting a heavy handbag.

2.2.2 Dexterous Hand Manipulation

Dexterous hand manipulation is a broad research area that has a variety of applications in computer graphics. Although a precise definition is still open to interpretation, dexterous hand manipulation is typically understood as the use of multiple fingers to achieve a desired object configuration. In computer graphics, researchers have shown that intricate control strategies, such as finger gaiting [136, 11], rolling/sliding [80, 93, 13], or grasping/regrasping [105, 73, 79, 138, 132], can be physically simulated on an anthropomorphic hand model. One of the most important assumptions for these dexterous hand manipulation techniques is that the manipulated objects are rigid bodies with only six degrees of freedom. Manipulating deformable objects is a more challenging problem due to more degrees of freedom and more complex collision phenomena.

In computer graphics, intensive research has been conducted in the area of cloth simulation [126, 15, 24, 42, 94, 95, 59, 130, 98, 97, 82]. Moreover, there has been a rich body of research on contact and collision for cloth simulation [23, 16, 48, 63, 101, 47, 125, 89, 9]. However, demo animations in cloth simulation are usually limited to a shirt lying on a mannequin or a handkerchief falling on an object. In such cases, the state of the cloth is only passively changing. There are few of techniques having an emphasis on controlling cloth motion [133, 128, 17, 49], but there is no work that utilizes dexterous hands to manipulate cloth. Automating dexterous hand manipulation of cloth is not an easy task due to the high dimensionality of cloth compared to the low dimensionality of control. Furthermore, there is a lack of accurate, efficient simulators that can simulate complex interaction between hands and cloth. In Chapter 5 and Chapter 6, we address the problem of synthesizing dexterous hand manipulation of cloth. We present a simulation technique to simulate realistic interaction between hands and cloth, as well as a physics based optimization technique to control cloth using hands.

2.3 Object Manipulation in Robotics

Full-body manipulation is extensively studied in robotics [46, 123, 137, 99]. Moreover, robotics researchers have explored kinematic and motor redundancy in operational space control to make robots achieve prioritized tasks goals [66, 96, 67, 113, 91]. However, due to the physical design and hardware constraints, existing robots only utilize predetermined manipulation strategies with intended manipulators. In Chapter 3, we present an algorithm built upon operational space control, and explore the problem of multitasking in full-body manipulation for animated characters.

Problems involving dexterous hand manipulation have also been frequently addressed in the context of robotics research. A wide range of manipulation strategies, such as grasping [92, 22, 104, 70], regrasping [127, 78, 65], finger gaiting [45, 68], finger pivoting [110, 108], and rolling/sliding [20, 44, 25, 35, 32], have been proposed for achieving different dexterous tasks. An alternative approach is to manipulate objects without grasping them, which is called nonprehensile manipulation. Nonprehensile manipulation allows using fewer actuated degrees of freedom to manipulate an object, increasing the set of reachable configurations of the object for a simple manipulator. A variety of strategies have been proposed, such as tumbling [112], tilting [38], pivoting [10], tapping [53, 52], two-pin manipulation [8], and two-palm manipulation [36, 139]. In Chapter 4, we present a method, which leverages a palm-like surface for dynamic nonprehensile manipulation and finger-like appendages to perform simple grasp and corrective control.

Researchers in robotics have also studied the problem of manipulating non-rigid objects such as fabric, cables, foam rubber, or sheet metal [71, 134, 39, 100, 19, 90, 102]. Many previous approaches enhance control and planning algorithms by using simulation techniques to estimate the state of the deformable objects. For example, robotics researchers used a cloth simulator to approximate the contours of clothes being folded by a PR2 robot [31]. Because the interaction between the robot hands

and the cloth was relatively simple (that is grasping only), their simulation applied position constraints to pin the cloth in the air instead of simulating the hands. In Chapter 5 and Chapter 6, we present algorithms that can simulate dexterous hand manipulation of cloth with higher motion fidelity.

CHAPTER III

SYNTHESIS OF CONCURRENT OBJECT MANIPULATION TASKS

In this chapter, we introduce a physics-based technique to synthesize human activities involving concurrent full-body manipulation of multiple objects. We design dynamic controllers to physically simulate upper-body manipulation and integrate it with procedurally generated locomotion and hand grasping motion. The output is a continuous animation of the character manipulating multiple objects and environment features concurrently at various locations in a constrained environment. To capture how humans deftly exploit different properties of body parts and objects for multitasking, we solve challenging planning and execution problems. We introduce a graph structure, a manipulation graph, to describe how each object can be manipulated using different strategies. The problem of manipulation planning can then be transformed to a standard graph traversal. To achieve the manipulation plan, our control algorithm optimally schedules and executes multiple tasks based on the dynamic space of the tasks and the state of the character. We introduce a task consistency metric to measure the physical feasibility of multitasking. Furthermore, we exploit the redundancy of control space to improve the character's ability to multitask.

3.1 Introduction

Performing multiple object manipulation tasks concurrently is an essential human activity in everyday environments. A mundane morning routine before going to work can involve numerous consecutive and concurrent tasks: picking up the briefcase on the floor, opening the refrigerator to fetch a lunch box, using the elbow to close

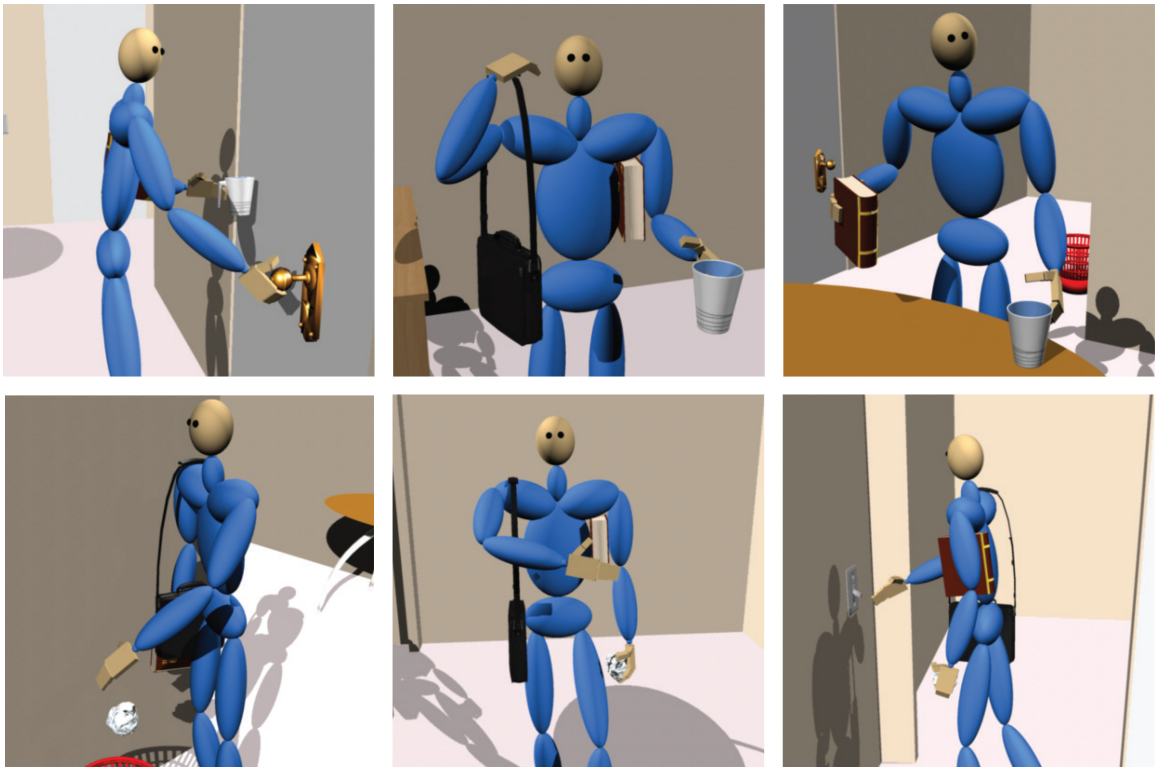


Figure 5: A simulated character manipulating multiple objects concurrently in different scenarios.

the refrigerator door, tucking the lunch box under the arm so the hand can search for keys in the pocket, and pushing the front door open by leaning on it. This sequence of tasks, which humans perform effortlessly, requires sophisticated planning and dynamic motion control, which have not been broadly explored in physics-based computer animation or robotics. Unlike existing robots, humans can employ a variety of manipulation strategies to interact with objects, such as using their hands, shoulders, elbows, torso, or even their head. Consequently, synthesizing full-body manipulation requires not only simulating physically-realistic joint motion, but also capturing how humans deftly exploit different properties of body parts and objects for multitasking.

In this chapter, we introduce a physics-based technique to synthesize human activities involving concurrent full-body manipulation of multiple objects. We view full-body manipulation as three interrelated layers of motor control: locomotion,

upper-body manipulation, and detailed hand manipulation. This chapter focuses on the second layer – we design dynamic controllers to physically simulate upper-body manipulation and integrate it with procedurally generated locomotion and hand manipulation. The main algorithm must overcome major challenges in both planning and execution.

Planning a valid sequence of manipulation strategies for a character is difficult because humans have abundant choices for manipulating an object. To circumvent this issue, our key insight is that, instead of making plans for each end-effector or body part of the character, we make plans for each object. We introduce a graph structure, a *manipulation graph*, to describe how each object can be manipulated by different strategies. An object's manipulation graph is based on its properties and contains a set of nodes, each of which represents a manipulation strategy (for example left hand, left shoulder, etc). An edge between two nodes represents a valid transition between two manipulation strategies (for example transporting the object from the left hand to the left shoulder). With the manipulation graph representation, the problem of manipulation planning can be conveniently transformed to a standard graph traversal.

Executing the manipulation plan has its own challenges. Because humans tend to act on tasks concurrently to save time or minimize traveling distance, a successful control algorithm must appropriately schedule multiple tasks, that is when to overlap tasks and when to execute them in succession. Given multiple tasks, we introduce a "task consistency" metric to measure the physical feasibility of multitasking based on the task spaces and the state of the character. Using this metric, we formulate a convex optimization to determine when and how to optimally overlap tasks. Furthermore, the dynamic controller must be general and robust to execute arbitrary multiple tasks simultaneously without interfering with each other. Inspired by the framework of operational space control [66], we exploit the redundancy of control

space to improve the character's ability to multitask. Specifically, our algorithm actively solves for an ideal next pose such that the task space in the next time step is most consistent with desired tasks. As a result, the character will try its best to achieve the current tasks while adjusting its motion continuously to improve the "multitasking consistency" for future tasks.

3.2 Related Work

As we discuss in Chapter 2, most previous work for synthesizing full-body manipulation exploited inverse kinematics and motion planning techniques to generate motion that satisfies desired manipulation tasks [83, 135, 69, 62, 61, 114]. In this chapter we focus on multitasking in full-body manipulation that exploits various manipulation strategies concurrently. In addition, we take the approach of physical simulation instead of kinematic procedures or motion capture, so that the motion can have more physical realism. Full-body manipulation is also extensively studied in robotics [46, 123, 137, 99]. However, due to the physical design and hardware constraints, existing robots only use predetermined manipulation strategies with intended manipulators. Consequently, concurrent manipulation scenarios described in this chapter have not been broadly explored in robotics.

The idea of associating interaction information with objects rather than characters has been proposed previously [107, 60]. The *smart object* approach stores manipulation information, such as grasping animation or approaching direction, as part of the object description. The interaction information is also useful for behavior modeling. Lamarche and Donikian [74] introduced a behavior representation that considers the involved body parts using concurrent state machines. As a result, the character can mix different behaviors without the need of creating specific behaviors exhaustively. The manipulation graph in our work is inspired by this similar idea. However, our

object representation does not contain any keyframes or animation sequences because the output motion is entirely physically simulated. The object representation, instead, stores the information of manipulation strategies and their transitions.

Our work also builds upon operational space control in robotics [66, 96, 67, 113, 91]. Operational space control exploits kinematic and motor redundancy to achieve prioritized task goals. These works demonstrated that humanoid robots could accurately track lower priority movement postures without interfering with the higher priority manipulation tasks. In computer animation, Abe and Popović [7] extended this framework to handle closed-loop joint structures. De Lasa et al. [33, 34] introduced an optimization scheme for task-space control, in which a nested sequence of objectives are optimized so as not to conflict with higher-priority objectives. The problem addressed in this chapter also depends on prioritizing multiple tasks, but we introduce two new ideas to handle concurrent manipulation tasks. By analyzing the subspace of task-equivalent control forces, we define a metric to measure the consistency of multiple tasks and schedule the tasks accordingly. Moreover, we actively optimize the character's next state so that the character is not only aiming to achieve the current tasks, but also adjusting its pose in preparation for future tasks.

Generating continuous locomotion is one of the most important applications in computer animation. Various kinematic techniques have been proposed and applied to gaming or virtual world applications [26, 109, 72, 28, 75]. To generate a long, continuous motion sequence from short motion clips, an effective motion blending technique must be able to handle walk cycles with different gait speeds, turning directions, stride lengths, and contact phases. We apply a similar idea as described by Kovar et al. [72] to interpolate similar motion clips while maintaining the correct contact states.

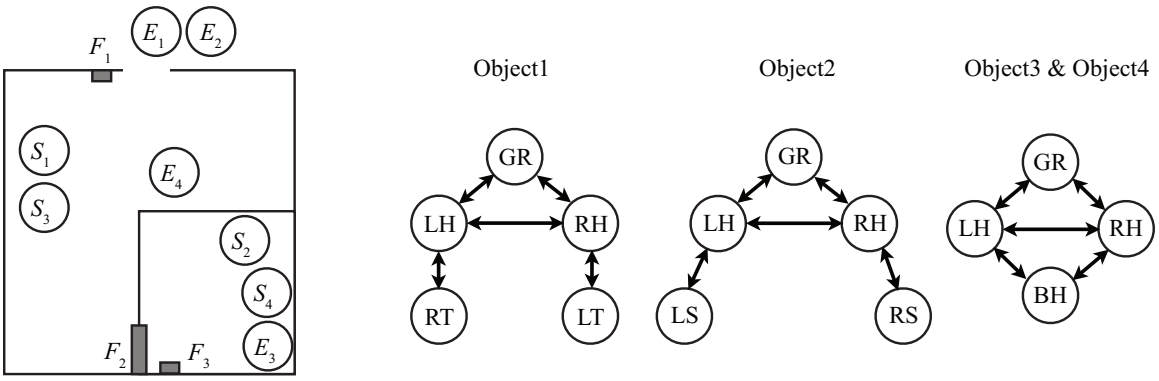


Figure 6: The input to our algorithm. Left: An environment map is a 2D illustration of a virtual environment. The user can specify an initial configuration (S_i) and a goal configuration (E_i) for each object i, as well as manipulatable features (F_i), such as doors or light switches. In this example, the user specified an environment map with nested spaces and three features, along with the tasks of transporting four objects. Right: A manipulation graph describes all possible strategies to manipulate an object. Here we show the manipulation graphs for four objects. A node in the graph represents one of the allowed manipulation strategies for this object. If two manipulation strategies can be executed in succession, we add an edge between their corresponding nodes. All nodes can transition to themselves but the edges are ignored for clarity. In this example, Object 1 is a book which can be picked up with either hand (LH/RH) from the ground (GR), and tucked under the left or right arm (LT/RT).

3.3 Overview

In this chapter, we introduce a physics-based method, which utilizes different manipulation strategies, to synthesize concurrent object manipulation. The input to our algorithm includes an environment map along with the information about the objects and features in the environment, and manipulation graphs that describe all possible strategies to hold, move, push, or release an object (Figure 6). The output of the algorithm is a continuous animation of the character manipulating multiple objects and environment features (for example doors or light switches) concurrently at various locations in a constrained environment.

The problem involves two stages: planning and execution (Figure 7). Given the user-specified input, the planning process produces a spatial path for locomotion and manipulation plans for all the objects and features in the scene. A manipulation plan

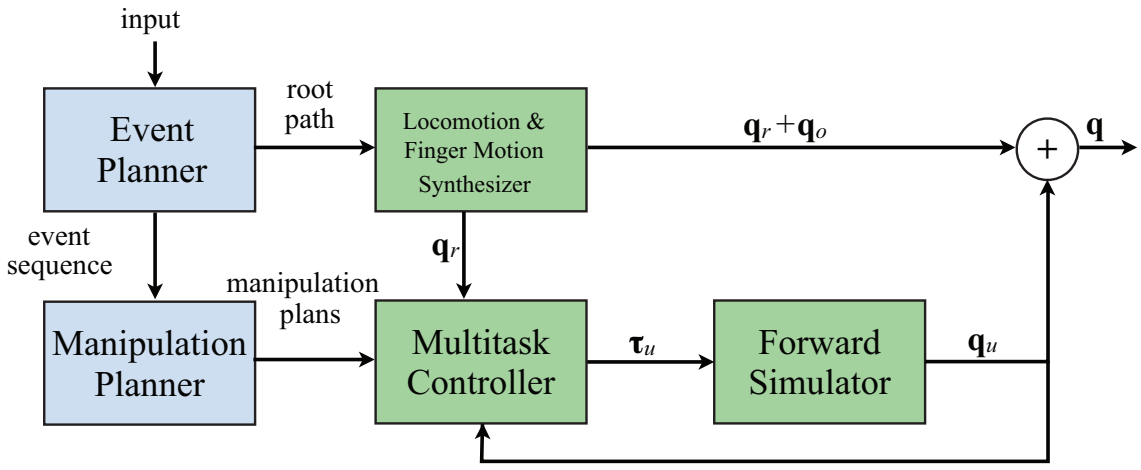


Figure 7: Algorithm overview. The problem involves two stages: planning (shown in blue) and execution (shown in green).

indicates which manipulation strategies should be used according to the manipulation graph of each object (Figure 9). During each execution step, the multitask controller determines a set of concurrent tasks and computes control forces $\boldsymbol{\tau}_u$ such that the concurrent tasks have minimal interference with each other. Finally, the forward simulator uses the control forces to simulate the next upper body pose $\mathbf{q}_u$ while the root motion $\mathbf{q}_r$, lower body and finger motion $\mathbf{q}_o$ are produced by a kinematic-based motion synthesizer.

3.4 Planning

The entry point of the algorithm is a two-step planning process, which consists of an event planner and a manipulation planner. Before we describe the planning algorithms, we first define a few terminologies in detail.

3.4.1 Definitions

An *environment map* is a 2D illustration of a virtual environment including walls, doors, furniture, and manipulatable features (F), such as door knobs or light switches

(Figure 6, Left). Each manipulatable feature F_i comes with a set of allowable manipulation strategies. For example, a door knob can only be manipulated using the left or right hand. On the environment map, the user can specify initial configuration (S_i) and goal configuration (E_i) for each object i to indicate the location and approaching orientation for pick-up and release of the object. We define an *event* as one of three actions: the pick-up of an object S_i, the release of an object E_i, or the interaction with a feature F_i, such as turning a light switch on or off.

In addition, a *manipulation graph* for each object needs to be specified by the user. A node in the graph represents one of the allowed manipulation strategies for this object. We define nine different types of strategies in this chapter: LH/RH/BH (use left/right/both hands to grasp the object), LS/RS (use left/right shoulders to carry the object), LT/RT (tuck the object between torso and left/right arm), and LE/RE (use left/right elbow to push the object). We also define a special node, GR, to indicate the "ground state" when the object is not manipulated by the character. If two manipulation strategies can be executed in succession, we add an edge between their corresponding nodes. For example, a book can be picked up by the left hand and tucked under the right arm. In this case, we add an edge between LH and RT (Figure 6, Right).

An *aggregate manipulation graph* combines all the input manipulation graphs into one (Figure 8(a)). Given n manipulation graphs, we construct each node of the aggregate graph by taking an n-tuple consisting of one node from each manipulation graph. Initially, the aggregate graph nodes consist of all possible n-tuples of object configurations. We prune nodes with configurations that cannot be achieved due to the mutual exclusivity of the manipulation strategies. For example, an aggregate node containing LH (left hand) and BH (both hands) is invalid if the character is not allowed to manipulate one object using the left hand and the other using both

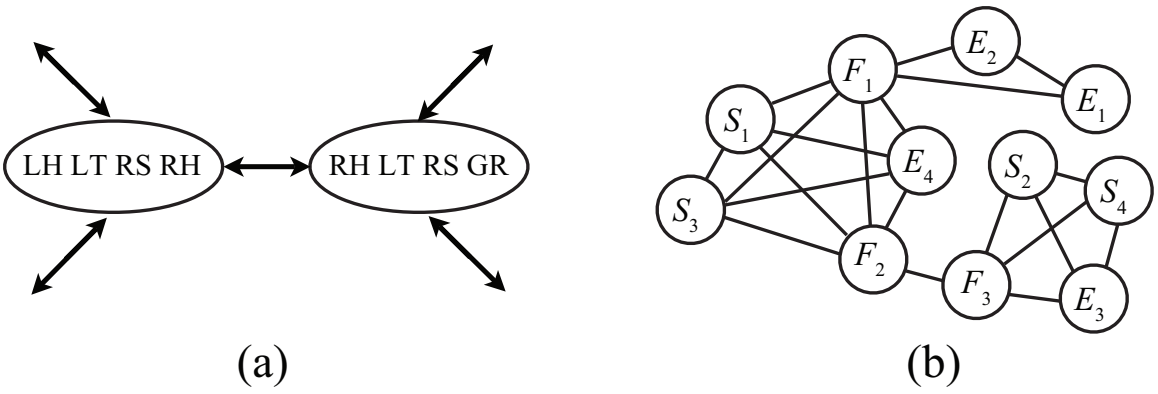


Figure 8: (a) Partial aggregate manipulation graph constructed from the four manipulation graphs in Figure 6. (b) The event graph for the scenario in Figure 6.

hands[1]. Once we define a valid set of aggregate nodes, we determine the connection of each pair of nodes based on the connectivity of the original manipulation graphs. Consider two aggregate nodes s_1 and s_2, where $s[i]$ denotes the manipulation strategy for object i in node s. We add an edge between s_1 and s_2 if and only if, for every object i, there exists an edge between $s_1[i]$ and $s_2[i]$ in its original manipulation graph.

3.4.2 Event Planner

The goal of event planner is to search for a valid event sequence which achieves all the required tasks on the environment map based on user-specified object configurations and initial feature states (for example a light switch is on or off). Our algorithm casts the search problem as a graph traversal. The first step is to construct a *event graph*, of which each node corresponds to an event (S_i, E_i, or F_i) on the environment map. If there is a collision-free path between two nodes on the environment map, we add an edge between them and assign the Euclidean distance as the cost of the edge. For example, Figure 8(b) shows that (S_1) and (S_2) cannot be connected directly because there is a wall between them.

[1]In this chapter, every manipulation strategy is mutually excluded with itself. In addition, BH is mutually excluded with LH and RH.

Before we can traverse the event graph, we need to identify two types of constraints. *Precedence constraints* enforce that all S_i nodes appear prior to their corresponding E_i nodes in the event sequence, because an object cannot be released before it is picked up. *Capacity constraints* make sure that currently "active" objects only employ manipulation strategies affordable by the character. We consider object i active if S_i is in the current path but E_i is not. To satisfy capacity constraints, the set of manipulation strategies employed by currently active objects must match one of the nodes in the aggregate manipulation graph.

We can now apply a standard depth-first-search (DFS) on the event graph to find a shortest path that visits every S node and E node exactly once, subject to precedence and capacity constraints. The output path is the optimal event sequence, $P = \{p_1, \cdots, p_n\}$ that achieves the required manipulation tasks. Based on a feature's state and description, we can remove its corresponding event from P if it does not require manipulation. For example, if a light switch is already on, (for example F_3 in Figure 6, Left) when the character enters the room, this feature event can be removed from the event sequence.

Since we know the location of each event in P, we can compute a spatial path for the root trajectory that visits every event in a sequence using P as a guide. In addition to reaching the event locations, the path must approach each event at the desired angle and avoid obstacles in the environment. Our algorithm first converts the given environment map to a distance map based on the locations of obstacles (for example walls and furniture). We then connect each pair of consecutive event locations with a Hermite curve, such that incoming tangents meet the desired approaching angles. If the curve intersects with an obstacle on the distance map, we move the point of deepest penetration to a collision-free location and subdivide the curve at that point. This step is repeated recursively until the curve is collision-free.

Event Sequence: S1 S3 F2 E3 S4 S2 F3 E4 F1 E2 E1

Event Constraints:

	S1	S3	F2	E3	S4	S2	F3	E4	F1	E2	E1
Obj1	LH/RH										LH/RH
Obj2						LH/RH				LH/RH	
Obj3		LH/RH		LH/RH							
Obj4					LH/RH			LH/RH			
Fea1									LH/RH		
Fea2			LH/RH								
Fea3							LH/RH				

Manipulation Plans:

	S1	S3	F2		E3		S4	S2		F3		E4		F1	E2		E1
Obj1	RH	RH	LT	LT	RH	RH	RH	LT	LT	LT	LT	RH	RH	RH	LH	LH	LH
Obj2									RH	RS	RS	RS	RS	RS	RH		
Obj3		LH	LH	LH	LH												
Obj4							LH	LH	LH	LH	LH	LH					
Fea1														LH			
Fea2				RH													
Fea3											RH						

Figure 9: The optimal event sequence, the event constraints and the derived manipulation plans for the scenario in Figure 6. Each row of the manipulation plans corresponds to an object and indicates the manipulation strategies planned for achieving the optimal event sequence.

3.4.3 Manipulation Planner

From the optimal event sequence, we can derive manipulation plans (Figure 9) by searching for a valid path, $Q = \{q_1, \cdots, q_n\}$, on the aggregate manipulation graph, where q_i represents the manipulation strategy associated with event p_i. Each event in the event sequence imposes certain constraints on the search problem. For example, if the event corresponds to pick-up or release of an object ($p_i = S_j$ or $p_i = E_j$), the manipulation strategy is constrained to left or right hand ($q_i[j]$ =LH or RH). If the event corresponds to a feature ($p_i = F_j$), the manipulation strategy must match one of the strategies allowed by that feature. The top table in Figure 9 shows the event

constraints for each object-event pair. The goal of the search algorithm is to fill in the manipulation strategies between S_i and E_i for each row O_i.

We apply DFS on the aggregate manipulation graph to find a valid path Q, subject to the event constraints. However, the manipulation plans resulting from Q can be unrealistic because they might contain too many "ground states" between the pick-up and release of an object, or transition between different strategies too frequently. As a result, the character's behavior may appear unnatural and unintelligent. To solve these issues, we prioritize the edges at each node during DFS, such that the adjacent nodes with no ground state and with the same manipulation strategies as the current node will be selected first. In addition, we allow the character more flexibility to rearrange the active objects before executing each event. For example, the character can tuck the book under its arm before opening the door. To achieve this relaxation, we allow the solution path to take an extra node before each q_i, that is adding another column before each S or E event and resulting a path $Q = \{q'_1, q_1, \cdots, q'_n, q_n\}$. Once a valid Q is found, we remove redundant nodes q'_i if $q'_i = q_i$.

3.4.4 From events to tasks

Before we can execute the actions specified by the manipulation plan, we need to translate the events in the manipulation plan to a set of concrete motor tasks. For example, a transition from LH to LS (Figure 6, Object 2) requires a task to transport the object from the left hand to the left shoulder, followed by another task to move the left hand back to a neutral position. For the examples shown in this chapter, four different tasks are defined: a tracking task, a holding task, a transporting task, and an attention task. We describe these tasks in further detail in Section 3.5.2.

The tasks associated with each event transition can be stored at the edge of a manipulation graph. We group the edges into three categories:

- L/RH $\rightarrow *$: A transition, from a hand to any manipulation strategy, associated

with a *transporting* task followed by a *tracking* task.

- $* \rightarrow$ L/RH: A transition, from any manipulation strategy to a hand, associated with a *tracking* task followed by a *transporting* task.

- $q \rightarrow q$: A self transition associated with a *holding* task.

We can now map the event transitions in the manipulation plan to a set of consecutive and concurrent motor tasks. In the next section, we will introduce an algorithm to execute these tasks efficiently.

3.5 Execution

The manipulation plan and the root path provide guidance for executing the character's motion. Our algorithm uses a forward simulator to physically simulate upper-body motion, while the root, lower-body, and finger motions are generated by kinematic procedures.

Given multiple tasks, humans tend to act on tasks concurrently to save time or minimize traveling distance. Therefore a successful controller for executing the manipulation plan needs to determine when and how to optimally overlap tasks. The task from the manipulation plan is defined by the desired position or orientation of a character's body node in Cartesian space. Due to the redundancy of the system, the control force that can achieve a particular task may not be unique. Inspired by the framework of operational space control [66], we explore the space of the task-equivalent control forces, and propose a control algorithm that optimally schedules and executes multiple tasks. Specifically, we present two new ideas to control the virtual character to imitate human multitasking. By analyzing the subspace of task-equivalent control forces, we introduce a "task consistency" metric to determine when to overlap tasks and when to execute them in succession. Furthermore, we exploit the relation between the character's pose and the space of task-equivalent control forces

to solve a desired pose, so that the character not only aims to achieve the current tasks, but also adjusts his pose to improve the "task consistency" for future tasks.

3.5.1 Multitask Controller

We first review the formulation of operational space dynamics and control. Let $\mathbf{q} \in \mathbb{R}^n$ be the independent degrees of freedom (DOFs) in the upper body of the character. The equations of motion in generalized coordinates can be expressed as follows:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{J}_p^T\mathbf{F}, \tag{7}$$

where $\mathbf{M}$ denotes the mass matrix, $\mathbf{C}$ includes Coriolis, centrifugal, and gravitational forces, $\mathbf{J}_p$ is the Jacobian matrix that projects external force $\mathbf{F}$ applied at a body point $\mathbf{p}$ from Cartesian to generalized coordinates, and $\boldsymbol{\tau}$ represents the generalized control forces applied internally by the character. If a task is to control the acceleration of a Cartesian point $\mathbf{x}$ on the character, it is more convenient to express the equations of motion also in the Cartesian space as follows:

$$\mathbf{J}\mathbf{M}^{-1}\boldsymbol{\tau} = \ddot{\mathbf{x}} + \mathbf{J}\mathbf{M}^{-1}\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \dot{\mathbf{J}}\dot{\mathbf{q}} - \mathbf{J}\mathbf{M}^{-1}\mathbf{J}_p^T\mathbf{F}. \tag{8}$$

Note that $\mathbf{J}$ is the Jacobian matrix evaluated at the point $\mathbf{x}$ and is in general different from $\mathbf{J}_p$. Equation 8 represents a simple linear relation between the commanding force $\ddot{\mathbf{x}}$ for the task and the required joint torques $\boldsymbol{\tau}$:

$$\mathbf{A}\boldsymbol{\tau} + \mathbf{b} = \ddot{\mathbf{x}}, \tag{9}$$

where $\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}$ and $\mathbf{b} = -\mathbf{A}\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{A}\mathbf{J}_p^T\mathbf{F}$.

Task scheduling. Using the linear relationship in Equation 9, we can define a task-inconsistency metric to measure the interference among multiple tasks. Let $\mathbf{P} \in \mathbb{R}^{n \times m}$ $(n > m)$ be the matrix whose range is the nullspace of $\mathbf{A}$. The torques that satisfy Equation 9 can be expressed as $\boldsymbol{\tau} = \boldsymbol{\tau}' + \mathbf{P}\mathbf{z}$, where $\boldsymbol{\tau}'$ is a particular

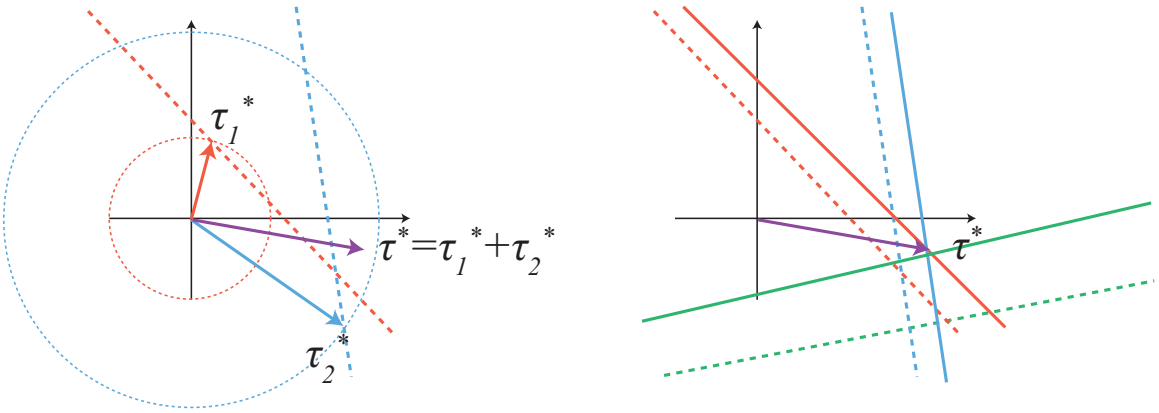


Figure 10: Left: Computation of optimal torque $\boldsymbol{\tau}^*$. Assuming there are two active tasks shown in red and blue, the dashed line indicates the torques that satisfy the task and the dashed circle indicates the torque bounds for the task. The optimal torque for the "red" task, shown as the red arrow, must lie on the red dashed line within the red circle, and be as parallel as possible to the blue dashed line. Similarly, the blue arrow indicates the torque that achieves the "blue" task while having the least interference with the red task. The final torque is the sum of these two individual torques shown as the purple arrow. Right: The optimal next pose. Changing the pose for the next time step effectively changes the future task spaces. Consider the two active tasks (red and blue dashed line) and a currently inactive task shown as a green dashed line. The optimal next pose will create new task spaces (solid lines) such that their intersection is closer to the current optimal torque (purple arrow).

solution of Equation 9 and $\mathbf{z}$ is an arbitrary vector in $\mathbb{R}^m$. Given two tasks T_i and T_j, a torque that satisfies T_i is considered consistent with T_j if it is in the range of $\mathbf{P}_j$. Therefore, we can define the most "multitask-able" torque for T_i as $\boldsymbol{\tau}_i^* = \boldsymbol{\tau}_i' + \mathbf{P}_i \mathbf{z}_i^*$, where $\mathbf{z}_i^*$ is the minimizer of the following convex optimization:

$$\begin{aligned} \mathbf{z}_i^* = \underset{\mathbf{z}_i}{\operatorname{argmin}} \quad & g(\mathbf{z}_i; \mathbf{P}_j) = \|\mathbb{P}_j(\boldsymbol{\tau}_i' + \mathbf{P}_i\mathbf{z}_i) - (\boldsymbol{\tau}_i' + \mathbf{P}_i\mathbf{z}_i)\|^2 \\ & \text{subject to} \quad \|\boldsymbol{\tau}_i' + \mathbf{P}_i\mathbf{z}_i\|_2 \leq d_i, \end{aligned} \tag{10}$$

where $\mathbb{P}_j = \mathbf{P}_j(\mathbf{P}_j^T\mathbf{P}_j)^{-1}\mathbf{P}_j^T$ denotes the projection matrix onto $\mathbf{P}_j$, and d_i defines the torque bounds for T_i. We choose to use l^2-norm to constrain the magnitude of the control torque so that the character does not use excessive torque for a task. The torque bounds d_i is set specifically for each task based on the rationale that the ideal torque may vary for different task types. The other option to set the torque bounds is to enforce torque limit for each individual joint according to the strength of the joint.

The inequality constraint in Equation 10 can be reformulated as $\mathbf{l} \leq \boldsymbol{\tau}_i' + \mathbf{P}_i \mathbf{z}_i \leq \mathbf{u}$, where $\mathbf{l}$ and $\mathbf{u}$ are torque bound vectors. Both options for the torque limit constraints can satisfy the convex optimization requirement. The residual of the optimization $r_i = g(\mathbf{z}_i^*; \mathbf{P}_j)$ measures how inconsistent T_i is with respect to T_j (Figure 10, Left).

When the character is dealing with a larger set of tasks, we simply replace $g(\mathbf{z}_i; \mathbf{P}_j)$ in Equation 10 with $g(\mathbf{z}_i; \bigcap_j \mathcal{R}(\mathbf{P}_j))$, where $\mathcal{R}(\mathbf{P})$ denotes the range of $\mathbf{P}$ and $\bigcap_j \mathcal{R}(\mathbf{P}_j)$ is the intersection of ranges of all tasks in the set except for task i. The residual of this optimization, $r_i = g(\mathbf{z}^*)$, indicates the inconsistency between T_i and the rest of the tasks. At each time step, the multitask controller computes the inconsistency metric r_i for every candidate task according to the manipulation plan. If the sum, $\bar{r} = \sum_i r_i$, is greater than a certain threshold, tasks are removed one by one until $\bar{r}$ is below the threshold. The order used to remove tasks is predefined based on task types: 1. Attention, 2. Tracking, 3. Transporting, 4. Holding. The remaining tasks constitute the active task set $\mathcal{A}^{(t)}$ for the current time step. The final optimal torque is computed as $\boldsymbol{\tau}^* = \sum_{i \in \mathcal{A}^{(t)}} \boldsymbol{\tau}_i^*$.

Optimal next pose. The algorithm described so far computes the optimal torque at each time step to best achieve currently activated tasks ($\mathcal{A}^{(t)}$), but it does not have any effect on the task space in the future. A more efficient way of multitasking requires the character to not only achieve the currently active tasks, but to anticipate other inactive candidate tasks. Because the task space parameters depend on the character's pose, that is both $\mathbf{A}$ and $\mathbf{b}$ are functions of $\mathbf{q}$, we can search for an ideal next pose which defines a task space more consistent with currently inactivate candidate tasks. In addition, the task space at the next time step should be similar to the current one so that the optimal torque computed by Equation 10 can be continuous over time.

To this end, our algorithm optimizes the pose $\mathbf{q}^{(t+1)}$ at the next time step such

that the intersection of all the candidate task spaces, which depends on $\mathbf{q}^{(t+1)}$, is brought closer to the currently optimal torque $\boldsymbol{\tau}^*$ (Figure 10, Right). We use a similar formulation as described in Section 3.5.1; the torques that achieve a task at the next time step must satisfy Equation 9, with $\mathbf{A}$ and $\mathbf{b}$ evaluated at $\mathbf{q}^{(t+1)}$. If there are multiple tasks at the next time step, we simply stack all the linear equations to obtain aggregate $\mathbf{A}$ and $\mathbf{b}$. The general solution for a torque that achieves all the tasks at next time step can be expressed as $\boldsymbol{\tau}(\mathbf{q}^{(t+1)}) = \boldsymbol{\tau}'(\mathbf{q}^{(t+1)}) + \mathbf{P}(\mathbf{q}^{(t+1)})\mathbf{z}$. Our algorithm optimizes the task space by finding a $\mathbf{q}^{(t+1)}$ such that the future intersection of task spaces is closer to the current optimal torque $\boldsymbol{\tau}^*$:

$$\underset{\mathbf{q}^{(t+1)},\,\mathbf{z}}{\operatorname{argmin}} \quad \|(\boldsymbol{\tau}'(\mathbf{q}^{(t+1)}) + \mathbf{P}(\mathbf{q}^{(t+1)})\mathbf{z}) - \boldsymbol{\tau}^*\|^2. \tag{11}$$

Because the optimal value for $\mathbf{z}$ can be expressed analytically as $\mathbf{z}^* = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T(\boldsymbol{\tau}^* - \boldsymbol{\tau}')$, Equation 11 can be rewritten as

$$\underset{\mathbf{q}^{(t+1)}}{\operatorname{argmin}} \quad \|(\mathbf{P}(\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T - \mathbf{I})(\boldsymbol{\tau}^* - \boldsymbol{\tau}')\|^2, \tag{12}$$

where optimization variables $\mathbf{q}^{(t+1)}$ are suppressed for clarity.

Once the optimal next pose, $\mathbf{q}^{*(t+1)}$, is computed, we still need to incorporate it into the current time step. We do so by exploiting redundancy in control space, as described in the next paragraph.

Prioritized task force. The final control force $\bar{\boldsymbol{\tau}}$ is the sum of multiple prioritized commanding forces. Using the operational space control framework ([67]), we resolve potential interference among tasks by projecting the secondary commanding forces $\boldsymbol{\tau}_s$ onto the nullspace of the primary task space: $\bar{\boldsymbol{\tau}} = \boldsymbol{\tau}_p + \mathbb{P}\boldsymbol{\tau}_s$, where $\boldsymbol{\tau}_p$ is the primary commanding force. In our formulation, the optimal torque $\boldsymbol{\tau}^*$ required to achieve currently active tasks is the primary commanding force. Tracking the optimal next pose $\mathbf{q}^{*(t+1)}$ is not essential for the current tasks, but it will make it easier to multitask in the future. Therefore, we track $\mathbf{q}^{*(t+1)}$ as a secondary task so that it

does not interfere with $\boldsymbol{\tau}^*$. In addition, we add a damping term for all the joints and make the damping force as a secondary task as well. The final commanding force can be written as

$$\bar{\boldsymbol{\tau}} = \boldsymbol{\tau}^* + \mathbb{P}(k_p(\mathbf{q}^{*(t+1)} - \mathbf{q}) - k_v\dot{\mathbf{q}}). \tag{13}$$

Additional forces. In addition to control forces, we also apply a gravity compensation force and a fictitious force to account for the effect of the lower body acceleration. We first compute the root acceleration, $\mathbf{a}_r$, by applying finite difference on the root positions generated by the locomotion synthesizer. The fictitious force, $-m_i\mathbf{a}_r$, is then added to each body node in the upper body, where m_i is the mass of body node i. Our simulation also considers the joint limits of the character. We compute the constraint force to enforce joint limits as a linear complementary problem.

3.5.2 Types of Tasks

This subsection provides the implementation details for the tasks we used to generate the examples in this chapter. Using the following formulation, we can compute a particular solution $\boldsymbol{\tau}'$ by solving Equation 9.

Tracking task: A task that moves a Cartesian point on the character toward a desired location $\bar{\mathbf{x}}$ in the world with desired speed $\bar{\mathbf{v}}$. We use a proportional-derivative (PD) controller to determine the commanding force: $\ddot{\mathbf{x}} = k_p(\bar{\mathbf{x}} - \mathbf{x}) + k_v(\bar{\mathbf{v}} - \dot{\mathbf{x}})$. A tracking task can also track the desired joint angle and joint velocity. In that case, the commanding force $\ddot{\mathbf{x}}$ represents the desired joint acceleration and $\mathbf{J}$ becomes an identity matrix.

Holding task: A task that maintains the current location of a Cartesian point $\mathbf{x}$ on the character against an external force $\mathbf{F}_x$ applied at $\mathbf{x}$. For example, if an object with mass m is held at $\mathbf{x}$, we set $\mathbf{b} = -\mathbf{AC}(\mathbf{q}, \dot{\mathbf{q}}) + \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{AJ}^T m\mathbf{g}$. In addition, we set the commanding force to: $\ddot{\mathbf{x}} = -k_v\dot{\mathbf{x}}$ to avoid non-zero velocity at $\mathbf{x}$. A holding task can also maintain the current orientation of a body point $\mathbf{x}$ against an external force.

Controlling the orientation can be done via a commanding torque $\dot{\omega} = -k_v\omega$, where ω is the angular velocity of the body node $\mathbf{x}$ resides. Equation 9 can be modified to: $\mathbf{A}\boldsymbol{\tau} + \mathbf{b} = \dot{\omega}$, where $\mathbf{A} = \mathbf{J}_\omega\mathbf{M}^{-1}$ and $\mathbf{b} = -\mathbf{AC}(\mathbf{q}, \dot{\mathbf{q}}) + \dot{\mathbf{J}}_\omega\dot{\mathbf{q}} + \mathbf{AJ}^T m\mathbf{g}$. The Jacobian $\mathbf{J}_\omega \in \mathbb{R}^{3\times n}$ relates the angular velocity of the Cartesian vector to joint velocity: $\omega = \mathbf{J}_\omega\dot{\mathbf{q}}$.

Transporting task: A task that combines the effort of holding and tracking to move an object to a desired location. We set $\ddot{\mathbf{x}} = k_p(\bar{\mathbf{x}}-\mathbf{x})-k_v\dot{\mathbf{x}}$ and $\mathbf{b} = -\mathbf{AC}(\mathbf{q}, \dot{\mathbf{q}})+ \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{AJ}^T m\mathbf{g}$. For both tracking and transporting tasks, we can gradually move the target point from the initial position to $\bar{\mathbf{x}}$ along a straight line with a bell-shape velocity profile to generate more natural human reaching motion.

Attention task: A task that controls the look-at direction of the character by setting the commanding torque as $\dot{\omega} = k_p\theta(\mathbf{v} \times \bar{\mathbf{v}}) - k_v\omega$, where $\mathbf{v}$ is the current look-at direction, $\bar{\mathbf{v}}$ is the target look-at direction, and θ is the angle between $\bar{\mathbf{v}}$ and $\mathbf{v}$. The attention task is initiated when the character starts to approach an object or an environment feature using one of the manipulation strategies. We define an attention zone as a sphere centered at the object of interest. A valid look-at direction is then defined as a vector from the location of the eyes to any point in the attention zone. Because real humans tend to look at the object carefully only at the beginning part of the reaching motion, we increase the radius of the attention zone as the character's manipulator gets closer to the object or the feature. This treatment introduces more overlap between attention zones of different tasks and allows the characters to manipulate multiple objects concurrently.

3.5.3 Locomotion and Finger Motion Synthesizer

We adopt existing work on motion blending and motion graphs to generate locomotion. A small set of mocap sequences containing a straight walk, single steps, and turning motions is used to create continuous walking sequences. We use the same

method described by Kovar et al. [72] to detect transition points in the dataset. Given the spatial path produced by the event planner, a sequence of walking cycles is selected and blended to follow the path. Additionally the locomotion synthesizer may refine its motion based on the proximity of active tasks. For example, if two events are very close to each other (for example E_3 and S_4 in Figure 6), the synthesizer may deviate from the original path and produce a small step toward the second one instead of a full walking and turning sequence.

The hand grasping motion is kinematically generated via a few keyframes and interpolation. When the hand is sufficiently close to grab the object, we stop simulating the object and rigidly attach the object to the hand. When the character releases the object, we resume the physical simulation on the object.

3.6 Results

We construct an articulated human character with 16 DOFs on the upper body and 18 DOFs on the lower body. The upper body motion is simulated using a rigid multibody simulator, DART [3]. We use a general optimization software, SNOPT [41], to solve for quadratic programs (Equation 10) and nonlinear programs (Equation 12). We create two examples to showcase the ability of the character to multitask in different scenarios: a baseline living room scenario and its variations, as well as a scenario in a coffee shop. The path planning and manipulation planning for the baseline living room example take 1.62 and 0.04 seconds respectively. On average, the simulation runs 3.5 times slower than real-time.

3.6.1 Living Room Example

Baseline scenario. Our baseline scenario is constructed from the example shown in Figure 6. According to the manipulation plans, the character picks up a book (S_1) with the right hand and a mug (S_3) with the left hand, and walks toward the bedroom while tucking the book under the left arm before opening the door with the

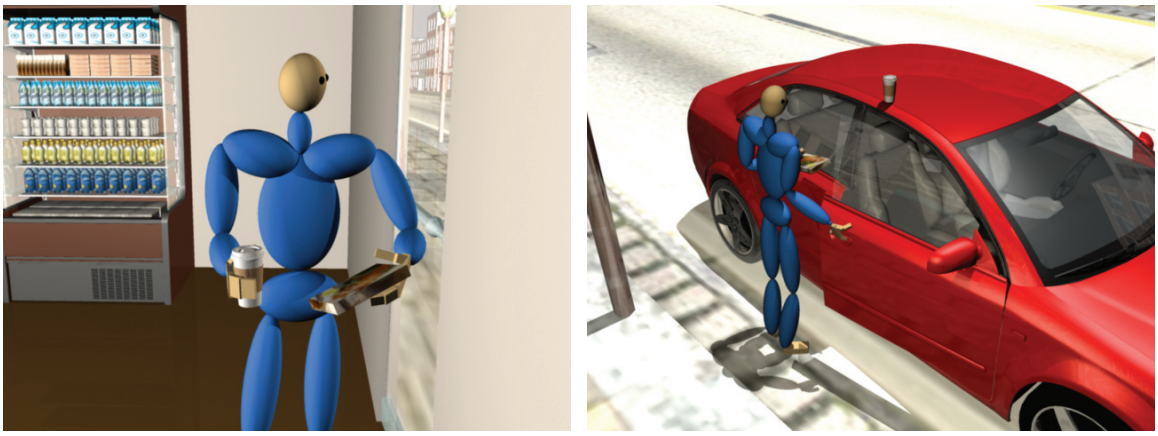


Figure 11: Simulated motion in the coffee shop example.

right hand (F_2). The character then moves the book back to the right hand before placing the mug on the table (E_3), grabs a crumpled paper using the left hand (S_4), tucks the book back under the left arm to allow the right hand to pick up a bookbag (S_2), puts the bookbag on the right shoulder and uses the right hand to turn off the light (F_3). After walking toward the corner of the living room while moving the book back to the right hand, the character drops the paper from the left hand (E_4) and walks toward the front door. Finally, the character turns off the switch using the left hand (F_1). This complex plan is computed automatically by our manipulation planner.

For this example, we modify the tracking task slightly for two occasions, reaching the torso and reaching the shoulder, to improve the aesthetics of the motion. Instead of setting a target position to $\bar{\mathbf{x}}$, we predefine a target trajectory such that the motion of the arm moves more naturally. Because both endpoints of the trajectory are determined in the character's local coordinates for these two special cases, we do not need to modify the trajectory for different objects or locations of the character.

Changing the environment map. We can modify any property on the environment map, delete or add objects, or change the manipulation graph for each object.

Our algorithm then automatically produces a new manipulation plan for the multi-task controller. In the example shown in Figure 6, we change the start configuration of the crumpled paper (S_4) and the end configuration of the mug (E_3). These two modifications drastically change the event sequence and manipulation plans. The motions for the living room example are illustrated in Figure 5.

3.6.2 Coffee Shop Example

In the second scenario, the character drops by a coffee shop to buy lunch. The manipulated objects include a cup of coffee and a lunch box, both of which have the same manipulation graph as Object 3 and Object 4 in Figure 6. In addition, we introduce two different types of doors in this scenario: a door that can be pushed open with an elbow and a car door that can only be opened by a hand (Figure 11).

After taking the coffee in his right hand, the character picks up a lunch box from the refrigerator using his left hand. When the character walks toward the door, our planning algorithm prefers to use his left elbow to push the door open instead of letting him put one of the items on the ground. Finally, when the character reaches his car outside, he cannot open the car door using any manipulation strategies except for LH or RH, which are both occupied by other objects. The only option left is to temporarily leaves one of the items on the closest surface (that is choosing GR strategy). In our example, the character chooses to put the coffee cup on the roof of the car.

3.6.3 Evaluations

Changing object properties. One advantage of using physics simulation to generate manipulation motion is that the character can react differently to objects with different physical properties. To demonstrate the effect of dynamics, we modify the physical properties of the object and compare the changes in the motion. In one example, as shown in Figure 12, the character tries to put the bookbag on the shoulder

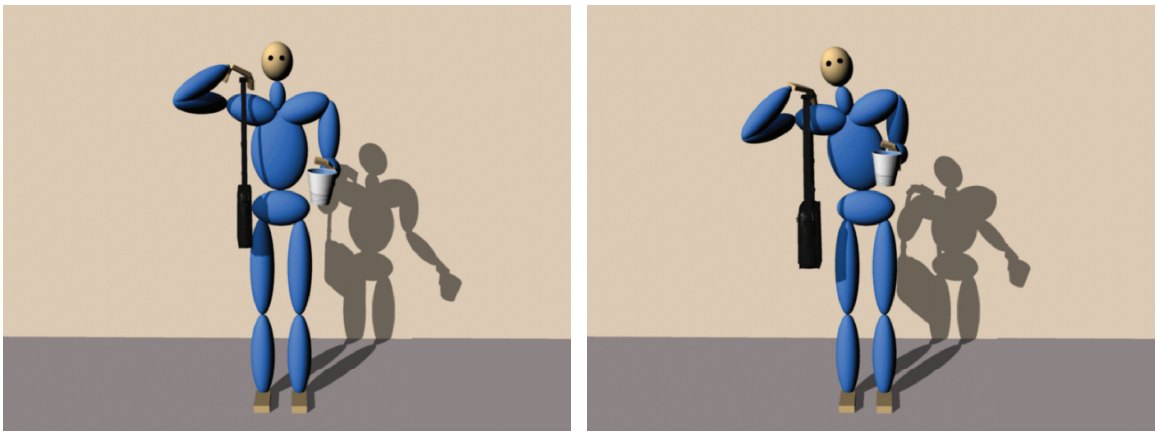


Figure 12: Comparison between heavy (left) and light (right) objects.

while holding a mug with the other hand. We show that the character leans further to the side when picking up a heavier bookbag due to dynamics, but still manages to maintain the upright orientation of the mug. In reality, it takes less effort to pick up a heavy object if we lean away from the object. To simulate this effect, our optimization will need to have an additional term that minimizes joint torque usage.

Comparison with a pose tracking approach. One simple method for generating multitasking motion is to apply inverse kinematics (IK) to solve for a target pose that satisfies multiple Cartesian constraints, and use a PD control scheme to track the target pose. This method may work in some situations, but it has a few drawbacks compared to our method. First, the trajectory required to achieve the target pose highly depends on unintuitive parameters of the PD trackers, whereas the motion trajectory generated by our method depends on multitask consistency. Second, tracking a target pose alone does not take inactive candidate tasks into account. Our method, on the other hand, continuously adjusts the character's poses in anticipation of future tasks. Third, it is hard to produce a natural target pose using standard IK without exploiting many example poses, while our method does not require any upper body poses. We demonstrate the difference between our method and a pose tracking approach in an example shown in Figure 13 where the character tries to

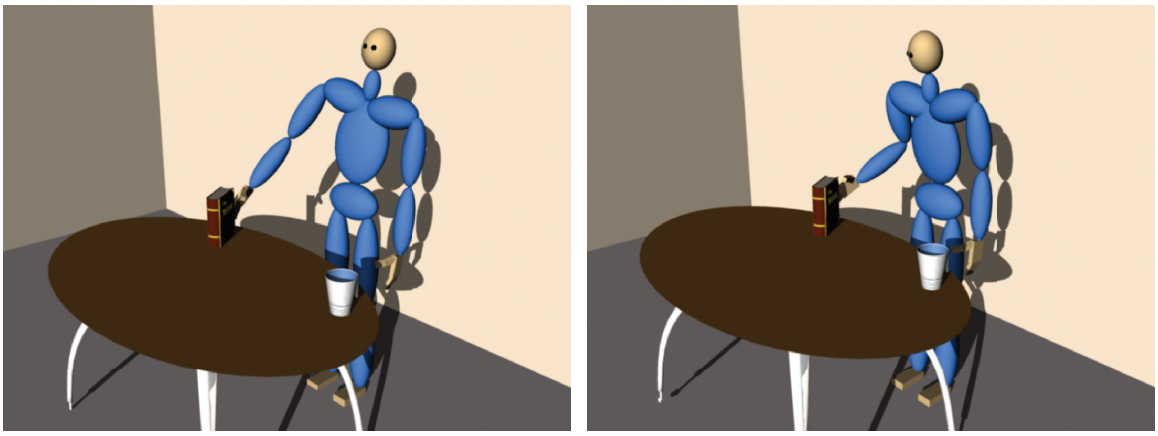


Figure 13: Comparison between a simple pose tracking approach (left) and our method (right).

reach two objects in sequence. The motion generated by pose tracking is clearly less natural than our result as shown in the video. Motion capture data could be used as guidance for tasks involving tracking. However, it is difficult to acquire appropriate mocap data in advance for all manipulation tasks and their combinations.

Optimal next pose. We demonstrate the effect of an optimal next pose using two challenging scenarios involving a few inconsistent tasks. In the first example as shown in the top row of Figure 14, the character tries to reach an object on a lower surface using the right hand while keeping the bookbag strap from sliding down the right shoulder. At the same time, the character must maintain the orientation of the mug in the left hand. Without the optimal next pose, the character can satisfy the tasks to maintain the orientation of the right shoulder and the left hand, but it fails to reach the object. On the other hand, with the optimal next pose, the character continuously adjusts its pose to lean toward the right side and eventually reaches the object. In the second example, the character tries to maintain the orientation of the mug in the left hand while tucking a book under the left arm using the right hand. Due to high inconsistency between these tasks, the orientation task eventually becomes inactive. Without the optimal next pose, the character completely ignored

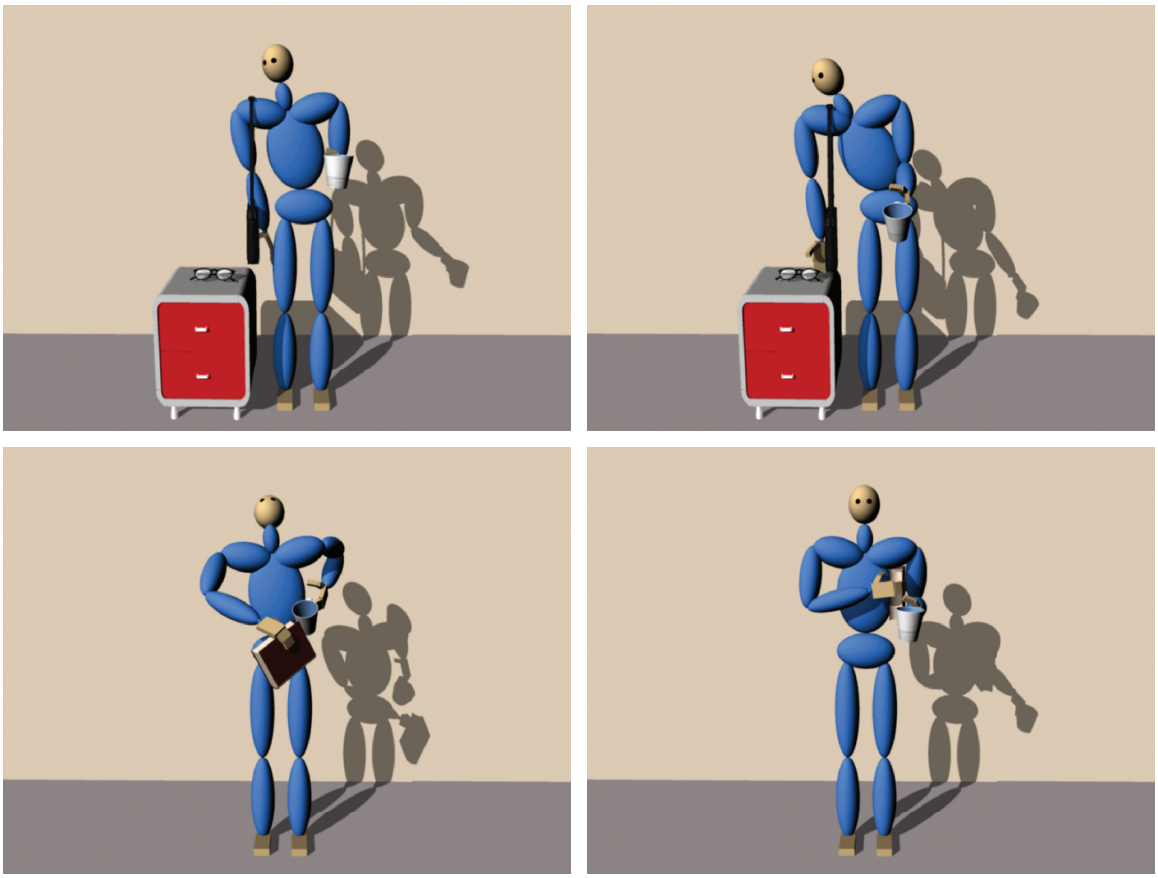


Figure 14: Top row: Comparison between the results with an optimal next pose (right) and without (left). Bottom row: Comparison between the use of a unified large torque bound for all tasks (left) and a proper torque bound for each individual task.

the orientation of the mug, resulting even greater inconsistency between these tasks.

Effect of task-specific torque bounds. As an alternative to our formulation for task scheduling, we can directly compute a torque vector which is the closest in Euclidean distance to the intersection of all active tasks (in Figure 10, the purple arrow would point at the intersection of the red and the blue lines). The drawback of this method is that we can only set a single torque bound for the aggregate torque that combines all the active tasks (that is the bound on the magnitude of the purple vector in Figure 10). In contrast, our method provides flexibility to define different torque bounds for different tasks, resulting in much more natural motion for multitasking.

We test the alternative method on the two examples used for testing the optimal next pose. In both examples, we find that it is difficult to determine a single torque bound for all active tasks. When the torque bound is too high, the character uses excessive torques to multitask, quickly leading to simulation blowup as shown in the bottom row of Figure 14. When the torque bound is too low, the character fails to achieve certain tasks that require a larger amount of torque. We define the torque bounds approximately based on the mass of the subtree rooted at each joint.

3.7 Limitations

Our current algorithm has a few limitations. First, we assume that the manipulation tasks are primarily done by the upper body and locomotion is done by the lower body. For simple pick-up and placement tasks that do not require much physical strength, this assumption can generate reasonable results. However, for more general whole-body manipulation tasks, such as pushing a heavy door or lifting a heavy object, coordination between locomotion and upper body manipulation is vital.

Our path planner implementation is very primitive and unable to handle extremely cluttered environments. Furthermore, we do not have a path planner at the level of upper body manipulation. When manipulating in a tight space, such as fetching an item in a packed refrigerator, the character's upper body is likely to collide with the environment or fail to move completely. To circumvent this issue, we plan to investigate a few broadly applied randomized algorithms proposed in previous work [76, 64].

Our algorithm only considers the shortest distance when planning the events. This is noticeable when we change the input of the baseline example, as the character carries both the mug and the book in and out of the smaller room before exiting, instead of picking them up on the way out. Other additional criteria, such as the amount of effort required for each task, could be taken into account during the DFS.

The walking motion in our examples can be largely improved if we use a larger mocap database and better motion editing algorithms. Our motion graph currently only contains 13 short clips.

3.8 Conclusion

In this chapter, we introduce a physics-based technique to synthesize human activities involving concurrent full-body manipulation of multiple objects. To capture how humans deftly exploit different properties of body parts and objects for multitasking, we solve challenging planning and execution problems.

We focus on the control problem of a single character walking through the constrained environment and interacting with the objects concurrently. A related topic that remains unexplored is to synthesize motions of multiple characters. In this case, manipulation tasks can change dynamically depending on the state of characters. For example, handing a object from one character to the other depends on the positions of two characters. Timing of multiple characters needs to be taken into consideration for both planning and execution stages to avoid collision and to synchronize tasks.

This chapter talks about the full-body manipulation motion. In the next chapter, we will explore dexterous hand manipulation, which is the other piece to complete a fully simulated virtual character for object manipulation. Instead of looking at the upper body motion with arms, we focus on hand motion with fingers.

CHAPTER IV

DEXTEROUS MANIPULATION USING BOTH PALM AND FINGERS

In this chapter, we present a technique to manipulate the orientation of an object using both palm and fingers of a virtual robotic dexterous hand. We formulate a simple algorithm to control the tilting angle of palm based on the conservation of mechanical energy and an empirical model of energy dissipation due to collisions. We demonstrate it can be applied to different types of objects rolling from an initial contacting face to a desired contacting face. Additionally, we develop a corrective controller for the fingers of the virtual robotic hand to improve the robustness against unexpected collision, irregular object, and noisy vision sensing input. The computation is simple and the controller can run in realtime on a virtual robotic hand. The experiments with the virtual Shadow Dexterous Hand model shows that the hand is able to pick up a given object on the table, to drop it on a specific spot on the palm, and to let it roll continuously and controllably on the palm, subject to the gravitational and contact forces.

4.1 Introduction

Using multifingered hands for dexterous tasks has many potential advantages. In addition to efficiency and versatility, multifingered hand dexterity provides additional degrees of freedom to increase the workspace of a manipulator [85]. On the other hand, not using fingers to grasp can also be an effective manipulation strategy in practice. For example, the absence of grippers largely simplifies the mechanical design, while increasing the flexibility to manipulate objects with various sizes and shapes [87].

While imitating anthropomorphic hands is arguably not an optimal solution to many practical applications, it is evident that a wide range of human manipulation tasks can benefit from integrative collaboration between appendages that emulate fingers and a surface that emulates the palm. In this chapter, we focus on the problem of manipulating the orientation of a polygonal object in hand. That is, given an initial orientation of the object on the table, can the virtual robotic hand pick up the object and re-orient it to a desired configuration in hand? Our approach leverages a palm-like surface for dynamic nonprehensile manipulation and finger-like appendages to perform simple grasp and corrective control. By integrating the use of a "palm" and "fingers", our approach is able to efficiently and robustly re-orient objects with different geometry and physical properties.

We propose a control technique for a virtual robotic hand to pick up a given object on the table, to drop it on a specific spot on the palm, and to let it roll continuously and controllably on the palm, subject to the gravitational and contact forces. We formulate a simple and fast algorithm to control the tilting angle of the palm based on the conservation of mechanical energy and an empirical model of energy dissipation due to collisions. While this nonprehensile approach requires minimal sensing and actuation capability, the object might not be executed precisely due to unexpected perturbations and inconsistency between the model and the real world. To mitigate execution errors, we develop another multifingers controller which corrects errors as the object rolls on the palm. Our method requires the geometry information of the object to be known a priori, but has no limitation on the convexity and symmetry of the shape, nor the location of the center of mass (COM).

The approach is demonstrated on a Shadow Dexterous Hand$^{\text{TM}}$ simulated using Gazebo simulator [4] with DART physics engine [3]. We show that the virtual robotic hand is able to manipulate a wide range of objects with different shapes, masses, moment of inertias, and friction coefficients, including nonconvex, irregular objects

with an offset center of mass. We also evaluate the proposed technique with noisy vision sensor input and objects with unsmooth surfaces.

4.2 Related Work

Dexterous hand manipulation uses anthropomorphic hand for precise manipulation tasks [85]. As we discuss in Chapter 2, problems involving dexterous manipulation have been frequently addressed in the context of robotics research. A wide range of manipulation strategies, such as finger gaiting [45], finger pivoting [110], rolling/sliding [20, 44, 25, 35], and regrasping [127], have been proposed for achieving different dexterous tasks. Although a precise definition of dexterous manipulation is still open to interpretation, many previous reviews provided nice discussion to summarize a variety of dexterous robotic systems based on their functionalities, hardware designs, and planning strategies. In particular, Bicchi [21] made a distinction between anthropomorphic hands to mimic the human anatomy and "minimalistic" hands to meet practical requirements, and argued for the necessity of hand dexterity. Ma and Dollar [85] argued that a simple gripper and a dexterous arm is sufficient for many applications, but a dexterous end-effector can increase the workspace of the arm. Instead of using a dexterous end-effector to make up for the limitations in arm functionality, our technique leverages nonprehensile manipulation on the palm to expand the possible motions of the object, while using multifingers mainly for the purpose of stable grasp.

Nonprehensile manipulation is to manipulate objects without grasping them. A variety of strategies have been proposed, such as tumbling [112], tilting [38], pivoting [10], tapping [53, 52], two-pin manipulation [8], and two-palm manipulation [36, 139]. The earlier work done by Erdmann et al. [37] solved for a sequence of tilting angles such that a polygonal object on the table can be orientated into a set of desired configurations. Later, Lynch and Mason [84] considered dynamic forces on the object

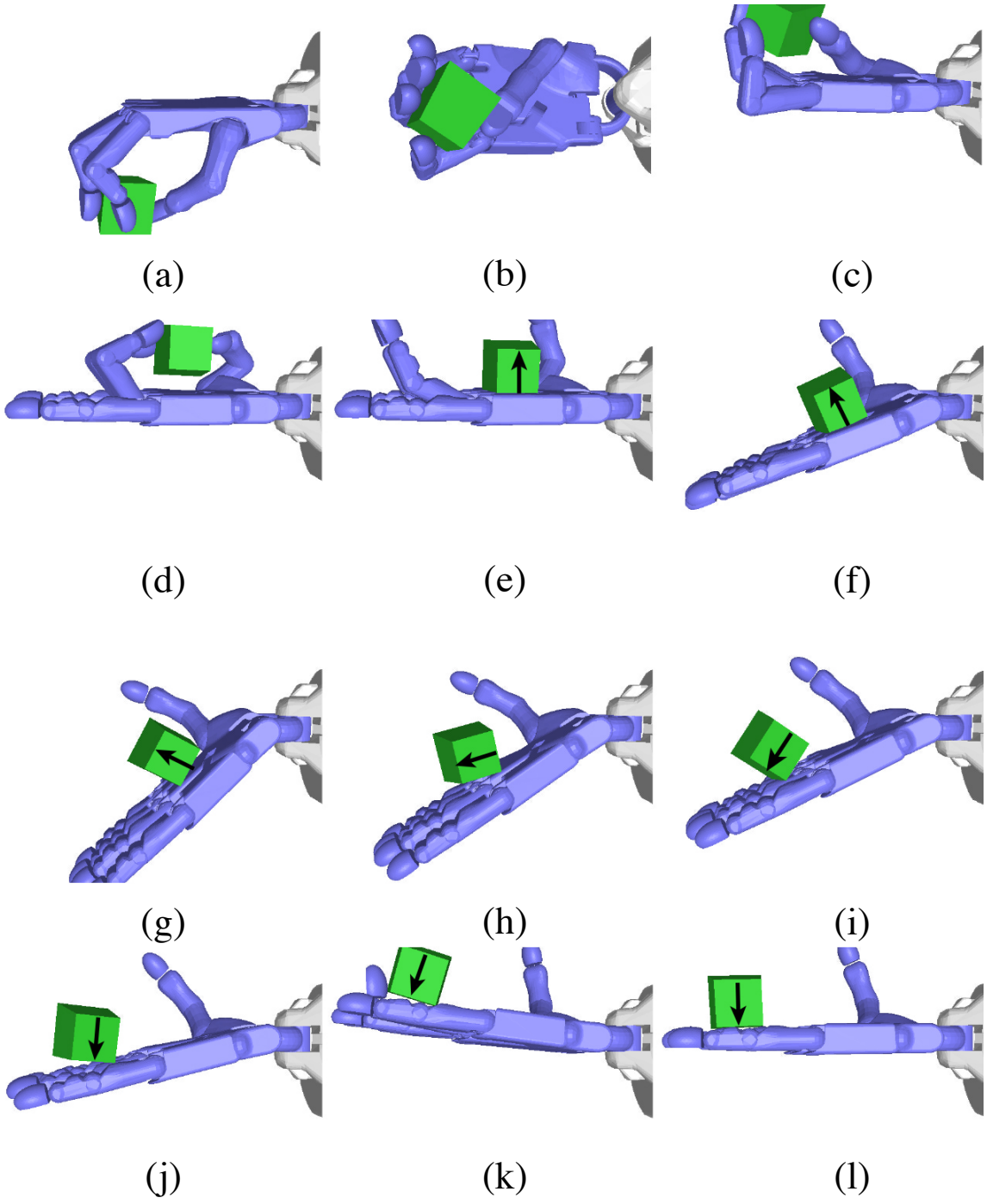


Figure 15: Grasp and roll. Screenshots from a simulated sequence of the Shadow Hand grasping and rolling an object to manipulate its orientation. The arrow represents the orientation of the object.

when planning the trajectories for the manipulator. By leveraging dynamics, they showed that an underactuated manipulator could snatch, roll, throw, and catch an object. Srinivasa et al. [117] tackled the problem of rolling a block sitting on the palm by 90 degree using a trajectory planning technique. They further extended the optimal trajectory to an optimal feedback controller using dynamic programming. This chapter addresses a similar nonprehensile manipulation problem. However, instead of planning the trajectory of the tilting angle using optimization techniques, we propose a different approach based on energy formulation to handle continuous rolling with multiple contacting faces.

4.3 Problem Statement and Assumptions

The problem we focus in this chapter is formulated as follows. Given a polygonal object rested on an arbitrary face on the table, the robotic hand must manipulate the object such that it rests on a desired face on the hand (Figure 15). We propose an approach in which a virtual robotic hand picks up the object from the table, drops it on the palm, and rolls the object to reach the desired contacting face while keeping the object in hand. Our approach utilizes both the palm for rolling the object and the fingers for grasping and correcting the orientation of the object.

Our algorithm makes the following assumptions:

1. The input object has a prism-like shape. That is, the object is a polyhedron with two polygonal bases joined by a set of parallel edges.

2. The prior knowledge about the object includes the position of the center of mass, the mass, the moment of inertia, and every vertex and edge expressed in the object frame.

3. The algorithm requires a vision sensor providing 3D coordinates of at least three vertices of the object in the world frame at all times.

4. The friction coefficient is sufficiently large such that the object does not slide on the virtual robotic hand.

4.4 The Palm

We first describe the algorithm to control an object rolling on the palm from an initial contacting face to a desired contacting face. The algorithm assumes that the object has been placed on the palm with its joining edges aligned with the x-axis of the palm (Figure 16). Because the object has the shape of a prism and its joining edges are perpendicular to the rolling direction, without loss of generality, we can reduce the 3D problem to rolling a 2D cross-section of the object on a plane. If the cross-section is concave, we simply take its convex hull and use it to represent the object.

The goal of the algorithm is to control the tilting angle of the palm (θ) such that the polygon can continuously roll across multiple edges and stop at the desired contacting edge. We break rolling motion to a sequence of contact cycles. Each contact cycle is associated with a contacting edge $\mathbf{e}$ between two vertices $\mathbf{r}_0$ and $\mathbf{r}_1$, which form a triangle with the center of mass $\mathbf{x}$. We define the length of the two edges of the triangle adjacent to $\mathbf{x}$ as d_0 and d_1, and the two angles adjacent to $\mathbf{e}$ as ϕ_0 and ϕ_1 (Figure 17).

An naïve approach considers each contact cycle individually and sets θ to be greater than $90-\phi_1$ for each contact cycle. Because the center of mass is not supported by the contact, the object will roll to the next contacting edge. However, this approach does not take into account the dynamics of the object and the gravitational force, resulting in a constantly accelerating rolling motion difficult to control and stop at the end.

We propose an algorithm that yields a more conservative rolling motion by considering the kinetic energy of the system. Our algorithm can be viewed as solving a sequence of inverted pendulum problems, each of which describes the motion of

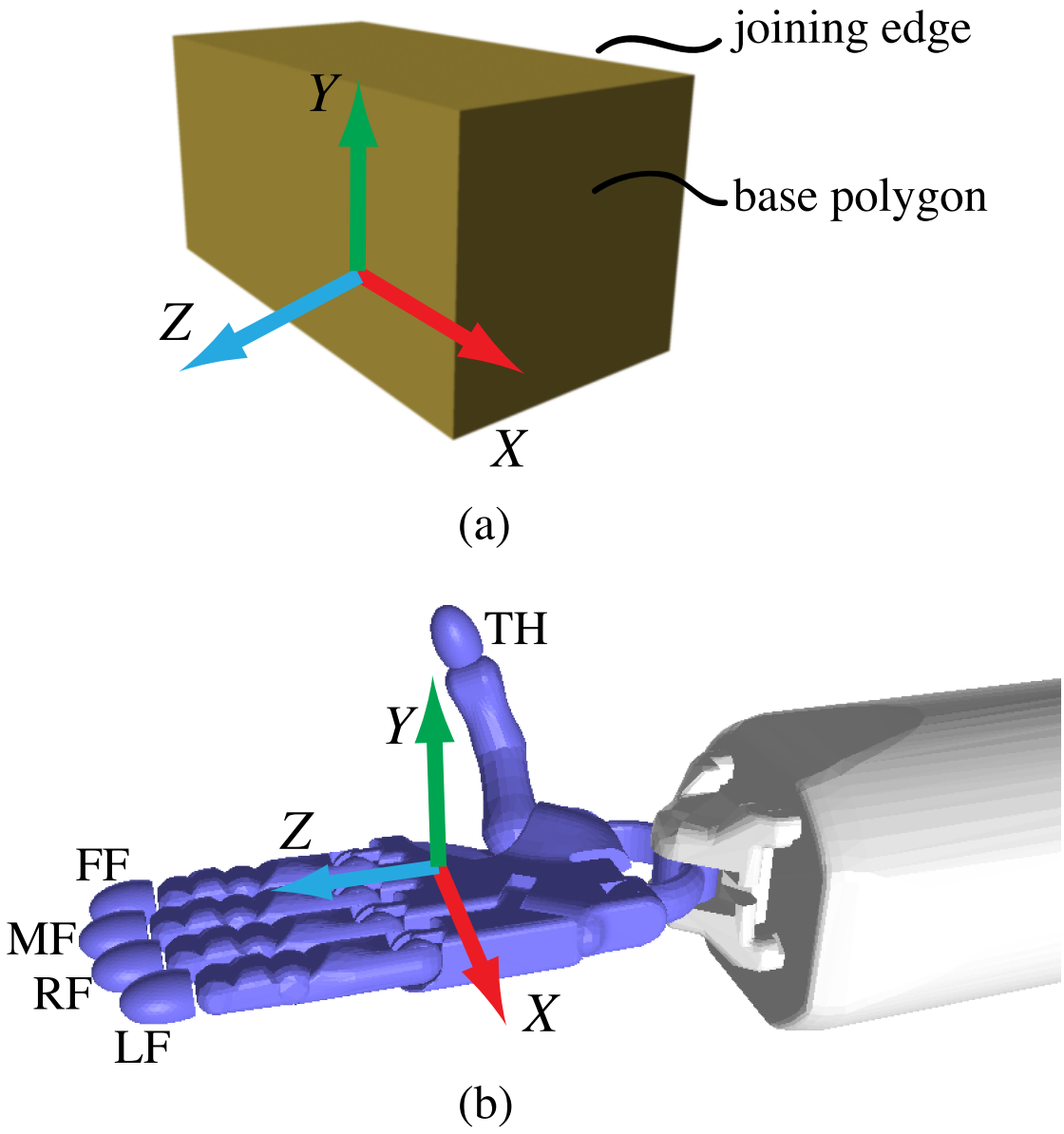


Figure 16: The object and the hand. (a) The object has a prism-like shape, which consists of two base polygons and a set of parallel joining edges. The object frame is illustrated in the figure. (b) The Shadow Dexterous Hand with 24 degrees of freedom. Five fingers are indicated by TH, FF, MF, RF, lF. The frame of the hand is illustrated in the figure.

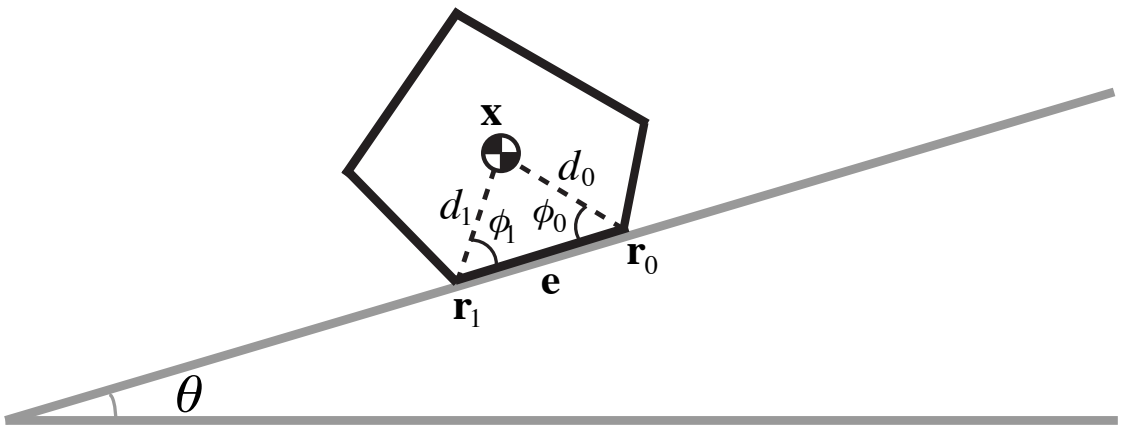


Figure 17: Notations for one contact cycle.

the center of mass of the polygon rotating about a vertex. If the pendulum at the apex has nonzero kinetic energy, the polygon will continue to roll to the next contacting edge. Based on this simple condition, we compute a sequence of θ to achieve continuous rolling to the desired contacting edge.

A contact cycle consists of three phases: dropping, colliding, and lifting (Figure 18). The dropping phase begins when $\mathbf{x}$ is at the apex and $\mathbf{r}_0$ is the contacting vertex. The colliding phase begins when $\mathbf{r}_1$ establishes contact with the hand. The lifting phase begins when $\mathbf{r}_0$ breaks the contact. When $\mathbf{x}$ reaches the next apex with $\mathbf{r}_1$ as the contacting vertex, the next contact cycle begins. We define the kinetic energy at a few key moments of a contact cycle as follows:

- E^0: The beginning of a contact cycle.
- E^-: The end of the dropping phase right before the collision.
- E^+: The moment after the collision and the beginning of the lifting phase.
- E^1: The end of the current contact cycle and the beginning of the next contact cycle.

During the dropping phase, the conservation of mechanical energy demands that

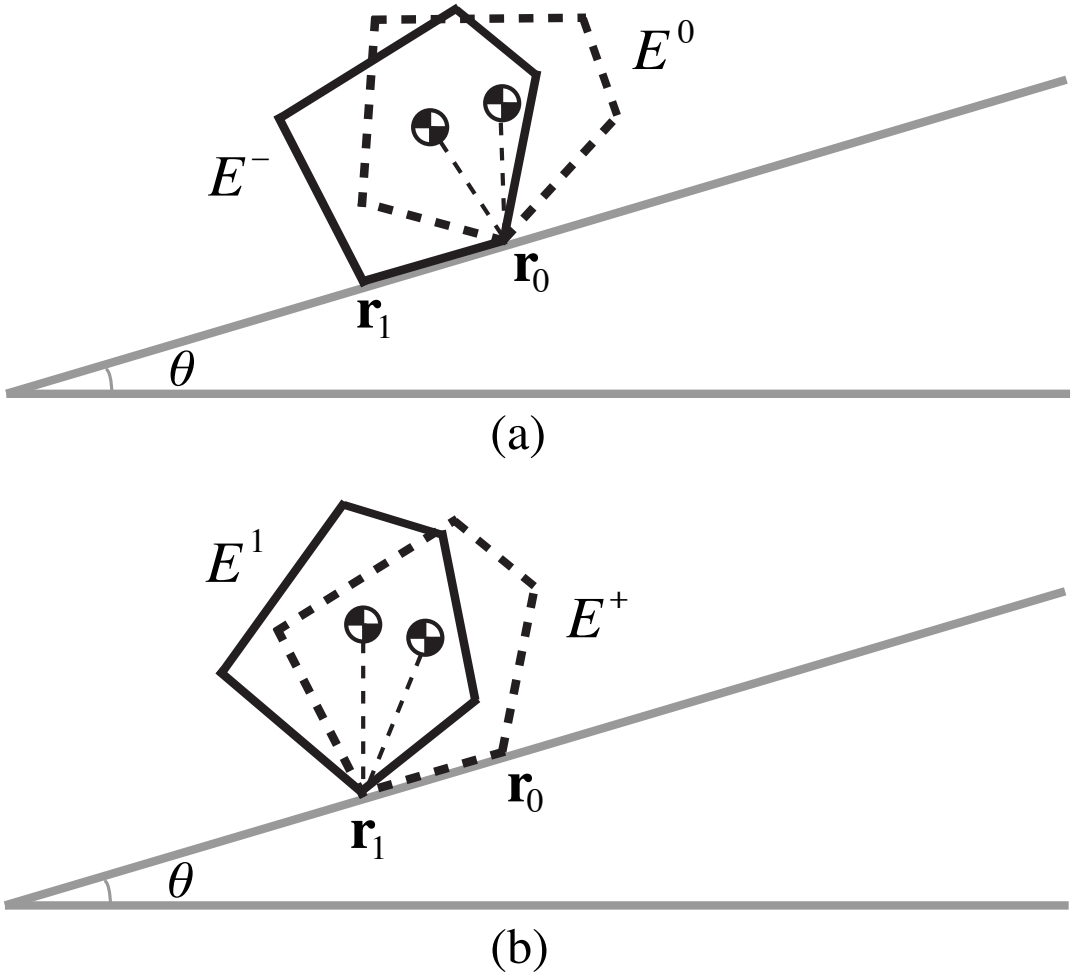


Figure 18: A contact cycle. (a) Dropping phase: The kinetic energy at the beginning (dashed figure) and end (solid figure) of the dropping phase are E^0 and E^- respectively. (b) Lifting phase: The kinetic energy at the beginning (dashed figure) and end (solid figure) of the lifting phase are E^+ and E^1 respectively.

the change of kinetic energy is equal to the change of potential energy:

$$E^- - E^0 = mgd_0(1 - \sin(\phi_0 - \theta)), \tag{14}$$

where mg is the gravitational force applied on the object.

The colliding phase models the dissipation of kinetic energy due to collision. We apply the empirical law for collision with a coefficient ϵ ($0 \leq \epsilon \leq 1$), which measures the kinetic energy dissipation. We set ϵ to 0.5 in our experiments:

$$E^+ = \epsilon E^-. \tag{15}$$

During the lifting phase, the polygon rolls about $\mathbf{r}_1$ until $\mathbf{x}$ reaches the apex, as the kinetic energy transforms into potential energy:

$$E^+ - E^1 = mgd_1(1 - \sin(\phi_1 + \theta)). \tag{16}$$

Using Equation 14, Equation 15, Equation 16, and the minimal kinetic energy condition $E^1 \geq 0$, the following inequality constraint on θ is derived for continuous rolling:

$$d_1 \sin(\phi_1 + \theta) - \epsilon d_0 \sin(\phi_0 - \theta) \geq d_1 - \epsilon d_0 - \frac{\epsilon}{mg} E^0. \tag{17}$$

The kinetic energy E^0 is computed based on the state of the polygon at the beginning of the contact cycle:

$$\frac{1}{2}(m\mathbf{v}^2 + \mathbf{I}\omega^2), \tag{18}$$

where $\mathbf{v}$ and ω are the linear and angular velocity of the polygon approximated by finite differencing the current and the previous positions.

Using angle transformation formulas, we rewrite Equation 17 as follows:

$$A \sin(\theta) + B \cos(\theta) \geq C, \tag{19a}$$

$$A = d_1 \cos(\phi_1) + \epsilon d_0 \cos(\phi_0), \tag{19b}$$

$$B = d_1 \sin(\phi_1) - \epsilon d_0 \sin(\phi_0), \tag{19c}$$

$$C = d_1 - \epsilon d_0 - \frac{\epsilon}{mg} E^0. \tag{19d}$$

We apply the rule for linear combination in trigonometry to obtain the following equation:

$$A\sin(\theta) + B\cos(\theta) = k\sin(\theta + \varphi), \tag{20}$$

where $k = \sqrt{A^2 + B^2}$, and φ is the unique angle satisfying following three conditions: 1) $-\pi < \varphi \leq \pi$; 2) $\sin(\varphi) = B/k$; 3) $\cos(\varphi) = A/k$.

The first and the last contact cycles are two special cases. For the last contact cycle, we simply negates the kinetic energy condition to $E^1 < 0$, which stops $\mathbf{x}$ from reaching the next apex after the object hits the desired contacting face. For the first contact cycle, we do not consider the dropping phase and colliding phase because both $\mathbf{r}_0$ and $\mathbf{r}_1$ are already in contact with the hand. The desired angle satisfies the minimal kinetic energy condition as in Equation 17 with E^+ replaced by the initial kinetic energy E^0 :

$$d_1\sin(\phi_1 + \theta) \geq d_1 - \frac{E^0}{mg}. \tag{21}$$

The algorithm solves for θ at the beginning of the contact cycle and commands the palm to achieve the new θ before the colliding phase starts. In theory, we need to adjust both the translation and the rotation of the wrist during the dropping phase, so that the palm reaches θ while $\mathbf{r}_0$ remains stationary in the world frame. In practice, however, we can directly set the wrist angle to θ without translating it because small motion at $\mathbf{r}_0$ has little impact on the rolling.

4.5 The Fingers

The rolling algorithm described in Section 4.4 utilizes passive dynamics so that the object can be manipulated by only the palm. To achieve manipulation robustly in a real scenario, however, the object needs to be first transported to the palm and the rolling motion sometimes needs to be corrected due to unexpected collisions (for example the palm or the object has a rough surface), inconsistency between the

assumed model and the real object, or noisy vision sensing input. We propose to use fingers to complement the palm for more versatile manipulation. In particular, we use fingers to grasp the object from the table, drop it on the palm, and provide corrective control to prevent object from overshooting or deviating from the plan.

We implement a simple grasp algorithm by controlling the pose of the hand and the force generated by the end-effectors. The algorithm first computes the desired contact points for each finger and applies the inverse kinematics method (IK) to generate a desired pose for the hand. The desired contact points can be in any locations on the object surface as long as they provide stable grasp and allow the joining edges of the object aligned with the x-axis of the hand when the object is dropped on the palm. We propose one possible way to achieve this goal: pick two opposing faces that are not base polygons, and place TH on one face and MF and RF on the other face. If FF and LF can reach the base polygons, we add additional contact points for more support. Once the object is in a stable grasp, the wrist of the robotic hand is commanded to turn 180° to a palm-up position.

To move the object towards the dropping location, we control the amount of force that fingers apply to the object. The total desired force $\bar{\mathbf{F}}$ and torque $\bar{\boldsymbol{\tau}}$ are determined by the deviation between the current object state and the dropping location and orientation through feedback equations:

$$\bar{\mathbf{F}} = k_p(\bar{\mathbf{u}} - \mathbf{u}), \tag{22}$$

$$\bar{\boldsymbol{\tau}} = k_o\alpha(\mathbf{v} \times \bar{\mathbf{v}}). \tag{23}$$

In this two equations, $\mathbf{u}$ and $\bar{\mathbf{u}}$ are the current and the desired positions of the object respectively, $\mathbf{v}$ and $\bar{\mathbf{v}}$ are the current and the desired directions of a vector fixed in the local coordinate frame of the object, α is the angle between $\mathbf{v}$ and $\bar{\mathbf{v}}$, and k_p and k_o are proportional gains for position and orientation. For n contact points on the object, we compute the desired contact force $\mathbf{f}_i$ at each contact point $\mathbf{p}_i$ by solving

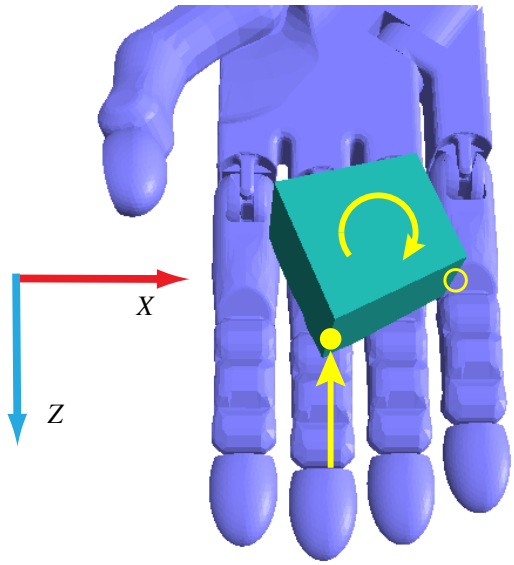


Figure 19: Angular deviation in the y-axis. A finger is used to create a torque on the object in the opposite direction of the deviation angle. In this case, the desired torque direction is indicated as the yellow arc arrow. The bottom left corner (shown as the solid yellow dot) is selected as the point of application on which a contact force (shown as the yellow arrow) will induce a torque in the desired direction. The closest finger, MF, is selected to provide the contact force.

the following optimization problem:

$$\min_{\mathbf{f}_1 \dots \mathbf{f}_n} \omega_1 \sum_{i=1}^{n} -\frac{\mathbf{f}_i}{\|\mathbf{f}_i\|} \cdot \mathbf{n}_i + \omega_2 \sum_{i=1}^{n} \|\mathbf{f}_i\|^2 \tag{24a}$$

$$\begin{pmatrix} \mathbf{I} & \cdots & \mathbf{I} \\ [\mathbf{p}_1 - \mathbf{x}]_\times & \cdots & [\mathbf{p}_n - \mathbf{x}]_\times \end{pmatrix} \begin{pmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_n \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{F}} \\ \bar{\boldsymbol{\tau}} \end{pmatrix}. \tag{24b}$$

The first term of the objective function minimizes the angle between the contact force and the contact normal, where $\mathbf{n}_i$ is the contact normal at the contact point $\mathbf{p}_i$. The second term minimizes the magnitude of the contact forces. We set the weight ω_1 and ω_2 to be 50 and 1 respectively. The equality constraint requires the total effect of contact forces equals to the desired force and torque. In the equality constraint, $\mathbf{I}$ is the 3×3 identity matrix, and $[\mathbf{p}_i - \mathbf{x}]_\times$ is the skew-symmetric matrix representing the cross product of the vector from the center of mass of the object, $\mathbf{x}$, to $\mathbf{p}_i$. We

use Jacobian transpose scheme [121] to control finger joints to exert desired contact forces:

$$\boldsymbol{\tau}_{int} = \sum_{i=1}^{n} \mathbf{J}_i^T \mathbf{f}_i, \tag{25}$$

where $\boldsymbol{\tau}_{int}$ indicates the control forces of the hand in generalized coordinates and $\mathbf{J}_i$ is the Jacobian matrix evaluated at $\mathbf{p}_i$. When the object reaches the dropping location, the fingers release the object at once. The dropping location is predefined in the coordinate frame of the hand.

Our algorithm also uses fingers to prevent overshooting and angular deviation about the y-axis of the hand. If an overshooting is detected at the last contact cycle ($E^+ > 0$), all four fingers are commanded to bend with a small angle (10°). To correct the angular deviation in the y-axis, we use fingers to create a contact force which induces a torque on the object in the opposite direction of the deviation angle. Depending on the desired direction of the torque, the algorithm selects one of the two extreme points in the x-axis as the point of application on the object (Figure 19). The closest finger to the selected point of application is commanded to bend forward until it strikes the object.

4.6 Results

We demonstrate our algorithm by simulating the Shadow Dexterous Hand manipulating a variety of objects in different scenarios. All the motions are simulated using the physics engine DART in Gazebo. DART is a multibody dynamic simulator formulated by Lagrange's equations in generalized coordinates. It handles collision and contact using an implicit time-stepping, velocity-based LCP (linear-complementarity problem) to guarantee non-penetration, directional friction, and approximated Coulomb friction cone conditions.

For all the results, the simulator integrates at 1000Hz (that is the integration

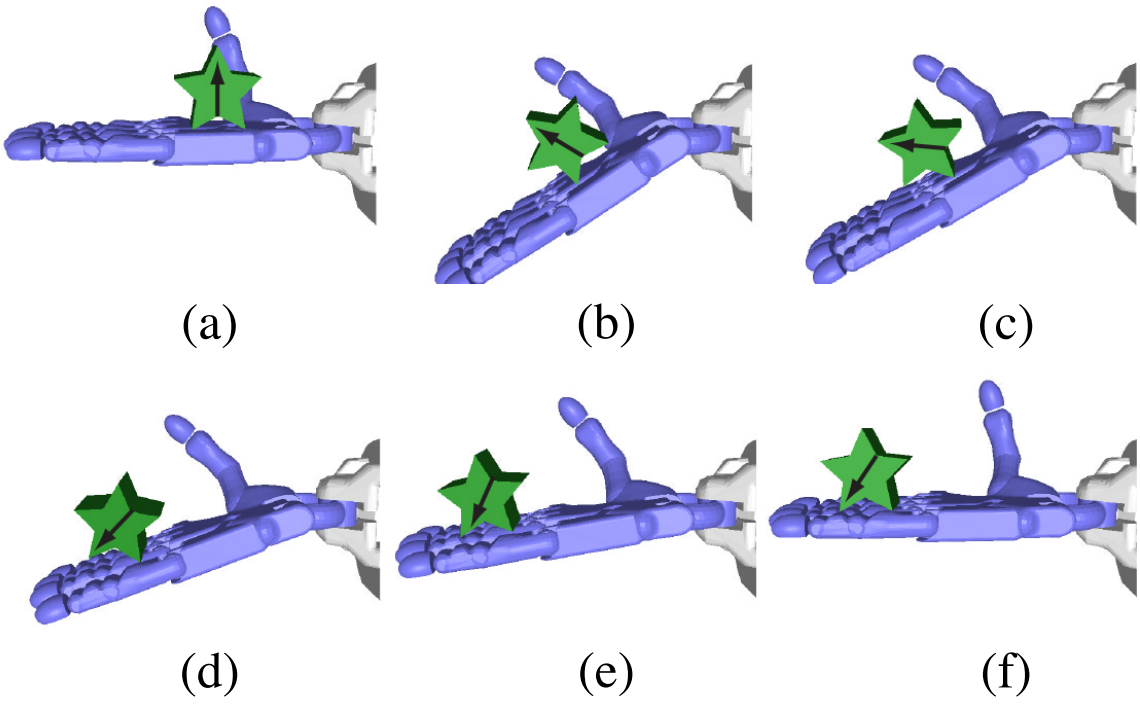


Figure 20: A nonconvex object. Screenshots from a simulated sequence of rolling a nonconvex object. The arrow represents the orientation of the object.

time step is 1 millisecond), but the controller is running at much lower frequency as it only sent one command to the robotic hand per contact cycle. We set the friction coefficient μ to 1.5 in all the results to prevent slippage. We also test smaller μ such as 1.0. While the titling angle is thus constrained to be less than 45°, it is successful for cases such as rolling the cube twice on the palm.

Our experiments show that the control algorithm is able to manipulate a variety of convex and nonconvex objects, such as a cube, a trapezoidal, or a star-shape prism (nonconvex base polygon, Figure 20). We also test the algorithm on objects with mass ranging from 0.1 kg to 1.0 kg, as well as objects with offset center of mass. Figure 21 shows the trajectory of θ when manipulating objects with different physical properties. In all cases, the algorithm is successful in rolling the object across multiple faces as desired, provided that the virtual robotic hand is longer than the required rolling distance. If the object is initially placed closer to finger tips, the same algorithm can

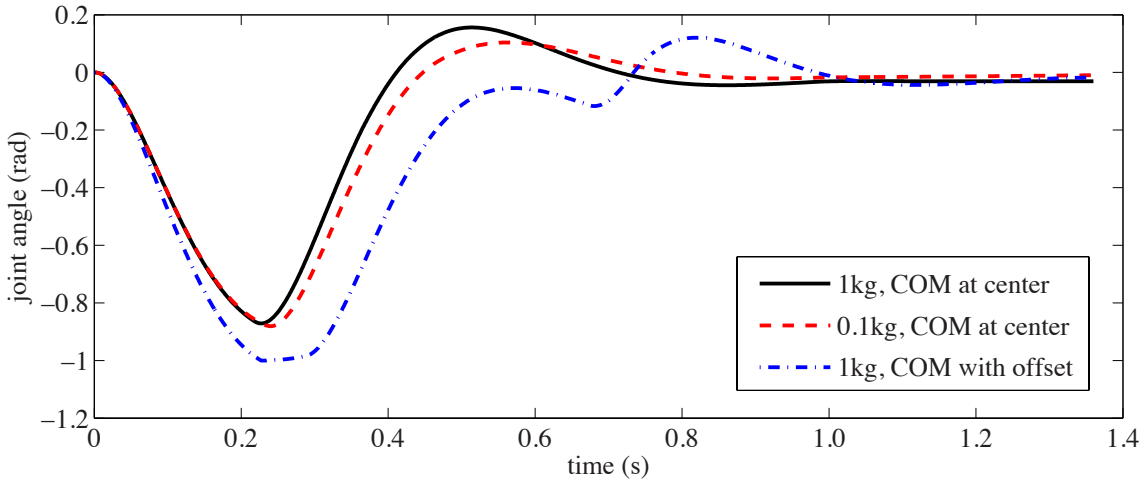


Figure 21: Trajectory of the wrist angle. Three sequences of rolling a cube across two contacting faces were simulated. Black line: $1kg$ cube with center of mass at the geometry center. Red dashed line: $0.1kg$ cube with center of mass at the geometry center. Blue dot line: $1kg$ cube with offset center of mass.

roll the object in the negative z-axis direction.

Two assumptions of the algorithm are relaxed during simulation. First, we allow the object to have slightly non-parallel joining edges (Figure 22 (a)). We also use objects with rough surfaces instead of analytical shapes considered by the algorithm (Figure 22 (b)). The results show that the violation of the assumption do not affect rolling significantly and the small errors can be corrected by the fingers. Second, we take into account the noise in the vision sensor input (Figure 22 (c)). To emulate the imperfect vision sensors in the simulation, we add Gaussian noise to the positions of vertices in the world frame and use the corrupted positions to approximate the frame of the object. With the variance of the noise being $5mm$, the successful rate of the control algorithm is still above 80%. More failure cases occur when we increase the noise. Most failure cases are due to the erroneous center of mass approximation which cause the palm to tilt too early or too late.

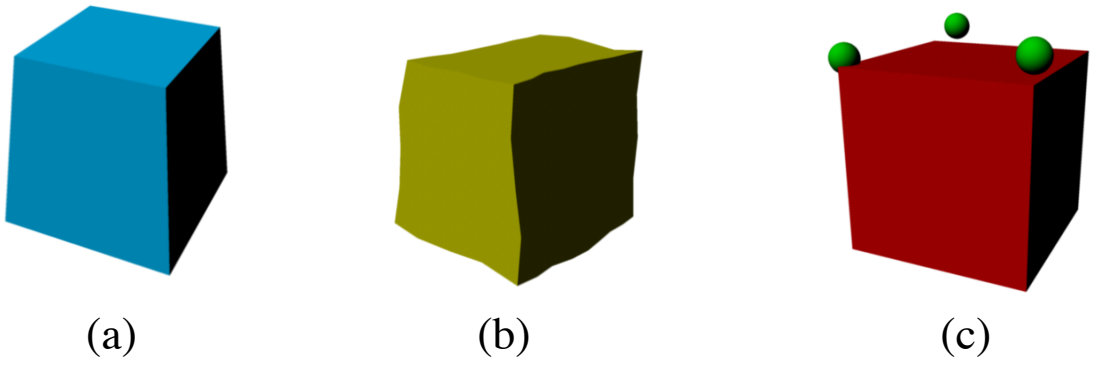


Figure 22: (a) A non-prism object of which the base polygons are not parallel. (b) An object with rough surfaces. (c) The green dots indicate the three vertices detected by an imperfect vision sensor.

4.7 Limitations

One limitation of our approach is that the detailed information about the object must be known in advance. This requirement could be problematic for applications when unknown objects need to be manipulated. Although we test the algorithm with non-prism and unsmooth shapes, the algorithm is still likely to fail on an object too different from a prism. For example, rolling a key on the palm would be a challenging case. Our algorithm cannot handle objects with curvy surface. Another limiting factor of our algorithm is that the wrist tilting angle is bound by the friction coefficient: $\mu \geq \tan(\theta)$, to prevent slippage.

4.8 Conclusion

In this chapter, we present a technique to manipulate the orientation of an object using both palm and fingers of a virtual robotic hand. We formulate a simple algorithm to control the tilting angle of palm and demonstrate it can be applied to different types of objects rolling from an initial contacting face to a desired contacting face. Additionally, we develop a corrective controller for fingers to improve the robustness against unexpected collision, irregular object, and noisy vision sensing input. The

computation is simple and the controller can run in realtime on a virtual robotic hand.

The proposed technique is general for virtual robotic hand with a palm-like flat surface and finger-like appendages. We expect that the same control algorithm can be applied to other virtual robotic hands. One future direction is to evaluate the algorithm on a physical system and utilize the experimental results to improve the simulation and the control algorithms.

This chapter talks about dexterous hand manipulation where the object is rigid. However, a large portion of the object manipulation tasks are related to non-rigid objects. Currently, there is no algorithm that can automatically create animation for dexterous hands manipulating cloth such as folding laundry. There are two challenges that make dexterous manipulation of cloth a hard problem. First, there is no simulator that can accurately and efficiently simulates the complex interaction between hands and cloth, due to the collision issues and the incorrect computation of contact forces. Second, the control of cloth is difficult since the number of controlled degrees of freedom is much less than the number of degrees of freedom of the cloth motion, especially when the control is constrained to be achieved by hands. In Chapter 5 and 6, we will address these two challenges in dexterous hand manipulation of cloth respectively.

CHAPTER V

COUPLING CLOTH AND RIGID BODIES FOR DEXTEROUS MANIPULATION

In this chapter, we introduce a simulation technique that couples cloth and rigid body simulation to synthesize dexterous manipulation of cloth. Our technique is built upon existing cloth and rigid body simulators. We focus on two main issues in current cloth simulators. First, the contact force computation does not take into account the dynamic information of the rigid body. Second, rigid body motion is unaware of the state of cloth, which frequently causes unsolvable collision situations. We develop a light-weight interface through which the rigid body and the cloth simulators communicate on a demand-driven manner to resolve these two issues. We demonstrate a set of basic manipulation skills, such as gripping, pinching, and pressing.

5.1 Introduction

Manipulating cloth is an important skill of daily living. From getting dressed, to folding laundry, many essential daily tasks depend on complex dynamic interaction between our hands and cloth. Unlike manipulation of rigid bodies, which at its heart is a problem of control and planning, synthesizing manipulation of cloth presents additional challenges in physics simulation. Currently there is a lack of an accurate, efficient, and unified simulator that simulates complex interaction between the hands and cloth.

Realistic cloth simulation has been extensively demonstrated in computer animation, but the interaction between cloth and rigid bodies is usually limited to simple cases, such as a shirt lying on a mannequin or a handkerchief falling on an object. In

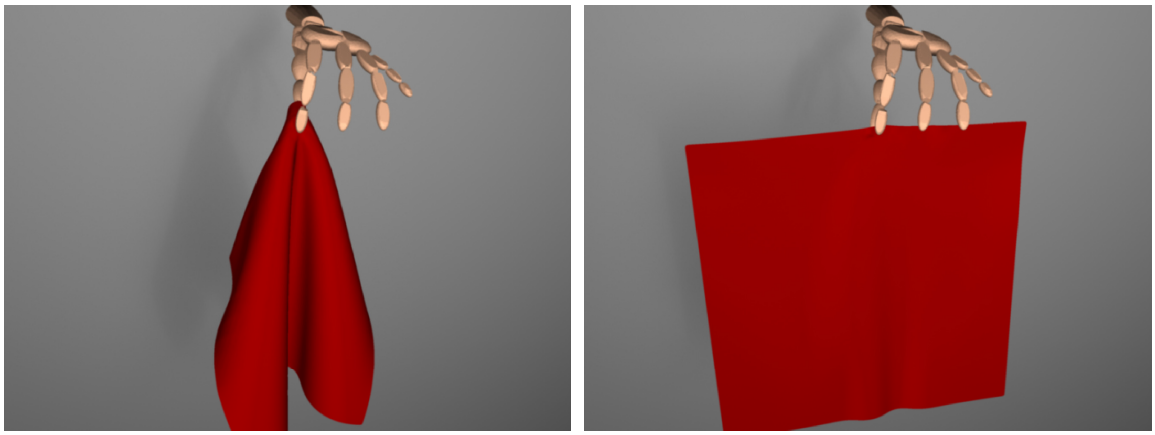


Figure 23: Pinching grasp. Left: A piece of cloth is held by the thumb and the index finger using our contact force computation. Right: The cloth slips out of the hand in ARCSim simulation.

current cloth simulators (for example Maya [5] or ARCSim [1]), the algorithm for coupling the cloth and the rigid body is not capable of simulating accurate interactions when rigid bodies impart forces to the cloth. As an illustrative example, current cloth simulators are unable to hold a piece of cloth, no matter how hard the hand grips the cloth (Figure 23). This example reveals an important issue of current simulators; the contact force computation only depends on kinematic information (position and velocity) of the rigid body, but dynamic information, such as inertial forces and other applied forces on the rigid body is ignored. A second important issue of current simulators is that rigid body motion is usually simulated or scripted separately from the cloth; the rigid bodies are oblivious to the existence of the cloth. In an example of two fingers pinching a piece of cloth, without the awareness of the cloth's current state, the space between two fingers might not be sufficient for the cloth simulator to resolve collisions, leading to unstable and unappealing cloth motion. This chapter introduces a new simulation technique to enable detailed dexterous manipulation of cloth, based on *pre-existing* cloth and rigid body simulators. Our algorithm facilitates communication between two simulators and computes crucial information for physical interaction. To illustrate how our technique improves upon current cloth simulators,

we characterize different coupling methods between the cloth and rigid bodies in the following table:

	$\mathbf{r} \rightarrow \mathbf{c}$ **None**	$\mathbf{r} \rightarrow \mathbf{c}$ **Kinematics**	$\mathbf{r} \rightarrow \mathbf{c}$ **Dynamics**
$\mathbf{c} \rightarrow \mathbf{r}$ **None**		ARCSim	
$\mathbf{c} \rightarrow \mathbf{r}$ **Kinematics**			Our work
$\mathbf{c} \rightarrow \mathbf{r}$ **Dynamics**			2-way couple

The columns indicate the information passing from the rigid body to the cloth (that is $\mathbf{r} \rightarrow \mathbf{c}$), while the rows indicate the information from the cloth to the rigid body (that is $\mathbf{c} \rightarrow \mathbf{r}$). To achieve accurate two-way coupling between two dynamic systems, the dynamic information needs to be passed on in both directions (that is bottom-right cell). Most state-of-the-art cloth simulators belong to the first row and the second column, because they only utilize the kinematic information of the rigid body and assume that the rigid body is unaware of the cloth. Our work strives to improve upon those cloth simulators. However, we argue that a true two-way coupling scenario is unnecessary for the applications of hand manipulation because the dynamic effects to the hand caused by the cloth are negligible. On the other hand, achieving correct two-way coupling involves redefining the state space and reformulating equations of motion, which requires a large amount of effort on reimplementation of existing simulators. Consequently, we believe that the "sweet-spot" for our applications is a coupling technique located at the second row and the third column. Our technique builds a light-weight rigid body interface that we call *rigid cloth patch*, which is a local representation of cloth in contact with a rigid body. Through rigid cloth patch, the rigid body and the cloth simulators communicate on a demand-driven manner in achieving two main goals: allowing rigid bodies to impart friction forces on cloth and avoiding unsolvable collision situations between the rigid bodies and

the cloth. We pass contact forces computed in the rigid body simulator to the cloth simulator so that proper friction forces can be applied. Conversely, we inform the rigid body simulator of the state of cloth by using dynamic rigid cloth patches to approximate the geometry of the cloth.

Our technique adds very little computational overhead and requires minimal modification of pre-existing simulators. Our implementation is built upon two open-source projects: ARCSim for cloth simulation and DART [3] for multibody simulation. We demonstrate a set of basic manipulation skills, including gripping, pinching, and pressing; that are frequently seen in daily activities such as dressing and folding clothes. Moreover, we compare our technique to the state-of-the-art cloth simulators and our technique shows favorable results.

5.2 Related Work

Previous work in dexterous hand manipulation in computer animation has demonstrated a variety of manipulation strategies on physically simulated hands, such as finger gaiting [136], rolling/sliding [80, 93, 13], or grasping/regrasping [105, 73, 79, 138, 132]. Researchers have also developed more accurate hand models with detailed simulation of tendons and muscles [120] and demonstrated their impact on control of manipulation tasks [111]. In spite of great progress made in this research area, existing techniques are limited to manipulating rigid objects with only six degrees of freedom. In contrast, robotics researchers have explored manipulation of deformable objects extensively, with an emphasis on folding clothes [100, 31, 19, 90]. Many previous approaches improved the control and planning algorithms by using cloth simulation techniques to estimate the state of the cloth. Because the interaction between the robot hands and the cloth is relatively simple (that is grasping only), the simulation method can be simplified to applying position constraints to pin the cloth in the air instead of simulating the hands. In this chapter, we aim to simulate the hands, the

cloth, and their effects to each other realistically. We show that with a more accurate simulation routine, a wide variety of manipulation strategies can be achieved.

Because our work is directly built upon the open source cloth simulator, ARCSim, we will describe a subset of research in cloth simulation that led to the techniques used by ARCSim. Since the seminal work by Terzopolous [126], many techniques have been proposed for cloth simulation [15, 95, 59, 82]. ARCSim adapts the elastic model formulation from several different approaches [24, 42, 94, 130] and used different algorithms for collision handling. There has been a rich body of research on contact and collision for cloth simulation [23, 16, 48, 63, 101, 47, 125, 89, 9]. The core algorithm for collision handling in ARCSim is based on Bridson et al. [23], who uses a geometry-based repulsive impulse to handle collision robustly. In addition, ARCSim improves the quality of the simulation by adapting inelastic projection for resolving simultaneous collisions [48]. ARCSim also takes advantage of remeshing techniques to improve computational speed and simulation quality. Narain et al. proposed an adaptive anisotropic remeshing technique to dynamically refine and coarsen triangle meshes in simulation [98, 97]. Though ARCSim is able to produce very compelling visual results with a relatively robust simulation process, it is not capable of simulating accurate interactions when a pair of rigid bodies impart forces on both sides of a piece of cloth. This is a crucial requirement for dexterous manipulation of cloth. Our technique enhances ARCSim by coupling rigid bodies and cloth in a more accurate fashion, while making minimal modification to its source code.

Two-way coupling of dynamic systems is an important research problem in computer animation. While graphics practitioners have demonstrated cloth-rigid interaction by treating a rigid body as a very rigid cloth using commercial cloth simulators (for example [5]), researchers have proposed various methods to handle two-way coupling between rigid bodies and deformable bodies [57, 116, 115, 101, 89]. Otaduy et al. [101] solved contacts between cloth and rigid bodies by implicitly solving a

large mixed linear complementarity problem (LCP). This requires accessing internal dynamic states of the simulators, such as the mass matrix, force vector and its derivative, and contact constraints. Our work treats existing cloth and rigid body simulators as black boxes without altering the internal formulation of collision handling. We solve a local nonlinear complementarity problem to compute contact forces. In conjunction with the proposed cloth rigid patches, our method is able to resolve the collision between cloth and rigid body in the pinched situation. Tackling the two-way coupling problem accurately requires formulating a unified set of equations of motion for both systems, and solving for the contact forces and the next state simultaneously. Given the requirements of our application, we chose a simpler and less involved approach to the coupling problem.

The physics-based simulation of interactions between hands and deformable objects can also be used in game applications and haptics research [56, 86]. Our method couples two dynamic systems without adding significant computation time to the simulation. If the cloth and rigid body simulators perform in real-time, the coupled system can also achieve real-time using our technique. Therefore, our technique has potential to be used in game applications.

5.3 Overview

Our method is independent from the integration scheme and contact algorithm used in the simulators. We utilize the contact force from the rigid body simulator to compute the cloth contact, and we take the state of the cloth in cloth simulator to create rigid cloth patches. Our algorithm is implemented with existing rigid body simulator DART, and cloth simulator ARCSim. Although we choose to implement our technique with these two simulators, we emphasize that our algorithm also applies to other physics engine, such as ODE [6] or Bullet [2], as long as the state of the rigid body, the state of the cloth, and rigid body contact forces can be accessed by the

APIs of the physics engine. The following pseudo code (Algorithm 1) illustrates the simulation procedure from the current state of the cloth $(\mathbf{x}^n, \mathbf{v}^n)$ (position and velocity) and of the rigid bodies $(\mathbf{q}^n, \dot{\mathbf{q}}^n)$ to the next state. We use $\mathbf{q}$ to represent the generalized coordinates of both the hand model and the rigid cloth patches, which are used to approximate the current state of the cloth (Section 5.5). The contributions of this chapter are highlighted in blue.

for *every time step* n **do**
  $\bar{\mathbf{v}}^{n+1} \leftarrow \text{ClothDynamics}(\mathbf{x}^n, \mathbf{v}^n)$;
  $\mathcal{V}_c, \mathcal{V}_r \leftarrow \text{ClothCollisionDetection}(\mathbf{x}^n, \bar{\mathbf{v}}^{n+1}, \mathbf{q}^n, \dot{\mathbf{q}}^n)$;
  for *every cloth vertex* i *in* $\mathcal{V}_c$ *or* $\mathcal{V}_r$ **do**
    if *in* $\mathcal{V}_r$ **then**
      $\mathbf{f}_N \leftarrow \text{RigidContactForce}(\mathbf{f}_c^n)$; // Section 5.4
    end
    else
      $\mathbf{f}_N \leftarrow \text{ClothCollision}(\mathbf{x}^n, \bar{\mathbf{v}}^{n+1})$;
    end
    $\tilde{\mathbf{v}}_i^{n+1} \leftarrow \text{FrictionForce}(\mathbf{x}^n, \bar{\mathbf{v}}^{n+1}, \mathbf{q}^n, \dot{\mathbf{q}}^n, \mathbf{f}_N)$; // Section 5.4
    $\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + \Delta t \tilde{\mathbf{v}}_i^{n+1}$;
    if $staticFriction =$ TRUE **then**
      $\mathbf{c}_i \leftarrow \text{PositionConstraint}(\tilde{\mathbf{x}}_i^{n+1})$;
    end
  end
  $\hat{\mathbf{x}}^{n+1}, \hat{\mathbf{v}}^{n+1} \leftarrow \text{StrainLimit}(\tilde{\mathbf{x}}^{n+1}, \mathbf{c})$;
  $\check{\mathbf{x}}^{n+1}, \check{\mathbf{v}}^{n+1} \leftarrow \text{InelasticProjection}(\hat{\mathbf{x}}^{n+1}, \hat{\mathbf{v}}^{n+1}, \mathbf{q}^n, \dot{\mathbf{q}}^n, \mathbf{c})$;
  $\mathbf{x}^{n+1}, \mathbf{v}^{n+1} \leftarrow \text{ClothRemeshing}(\check{\mathbf{x}}^{n+1}, \check{\mathbf{v}}^{n+1}, \mathbf{q}^n, \mathbf{c})$;
  $\bar{\mathbf{q}}^{n+1}, \dot{\bar{\mathbf{q}}}^{n+1} \leftarrow \text{UpdateRigidClothPatches}(\mathbf{x}^{n+1}, \mathbf{v}^{n+1}, \mathcal{V}_r)$; // Section 5.5
  $\boldsymbol{\tau} \leftarrow \text{ControlForce}(\bar{\mathbf{q}}^{n+1}, \dot{\bar{\mathbf{q}}}^{n+1})$; // Section 5.6
  $\mathbf{q}^{n+1}, \dot{\mathbf{q}}^{n+1}, \mathbf{f}_c^{n+1} \leftarrow \text{RigidSimulation}(\bar{\mathbf{q}}^{n+1}, \dot{\bar{\mathbf{q}}}^{n+1}, \boldsymbol{\tau})$;
end

Algorithm 1: Cloth and Rigid Simulation Algorithm

The time-stepping function begins with advancing the cloth to the candidate velocity $\bar{\mathbf{v}}^{n+1}$ by computing the cloth dynamics in the standard way [24, 42, 94, 130]. We then apply the collision detection algorithm in ARCSim to identify a set of cloth vertices $\mathcal{V}_r$ colliding with rigid bodies and another set $\mathcal{V}_c$ colliding with the cloth itself. The algorithm first determines the normal contact force $\mathbf{f}_N$ for each vertex.

If a cloth vertex i is in $\mathcal{V}_r$, we use the normal contact forces $\mathbf{f}_c^n$, computed by the rigid body simulator in the previous time step. Otherwise, we use the repulsive force computed by the cloth simulator to be its $\mathbf{f}_N$. Next, we compute the friction force based on Coulomb's friction model (Section 5.4). To ensure that the static contact points are not subject to changes made by the following refinement steps in the cloth simulator, that is strain limit, inelastic projection and remeshing, we enforce position constraints on those static vertices. After the cloth is advanced to its next state, we update the state of the rigid cloth patches to approximate the new state of the cloth (Section 5.5). Finally, we compute the control force for the hand (Section 5.6) and advance the rigid body simulator to the next time step.

5.4 Contact Force

Computing correct contact forces from the hand to the cloth is crucial for many dexterous manipulation applications. Most current cloth simulators (for example [1, 5]) compute contact forces based only on the position and the velocity of the rigid body. The consequence of an inaccurate contact force is particularly evident when the rigid body is not interpenetrating the cloth but is imparting force on the cloth (for example two fingers pinch-grasp a piece of cloth). We describe a new method for handling contact forces successfully in the above situations.

5.4.1 Normal Force

If a cloth vertex i is marked as being in contact with one or more rigid bodies (that is in the set $\mathcal{V}_r$), we compute its normal contact force using the collision handling routine in the rigid body simulator. DART solves the rigid body contact force as an implicit time-stepping, velocity-based LCP to guarantee non-penetration, directional friction, and approximated Coulomb friction cone conditions [118]. After performing collision handling in the rigid body simulation, we store a list of contact points for each rigid body.

For each rigid body that the cloth vertex $\mathbf{x}_i$ collides with, we find the nearest contact point $\mathbf{p}$ in the list of contacts for that rigid body and assign the normal contact force at $\mathbf{p}$ to $\mathbf{x}_i$. This normal contact force will be divided by the number of cloth vertices associated with $\mathbf{p}$, after all the cloth vertices have been visited. We store all the normal contact forces applied on $\mathbf{x}_i$ in a vector $\mathbf{f}_N \in \mathbb{R}^{3M}$, where M is the number of rigid bodies that $\mathbf{x}_i$ is in contact with. For those cloth vertices that are not in contact with any rigid body, we simply obtain the normal contact forces from the repulsive forces computed in the cloth simulator.

5.4.2 Friction Force

For every cloth vertex $\mathbf{x}_i$ in $\mathcal{V}_c$ or $\mathcal{V}_r$, our algorithm formulates a nonlinear complementarity problem to compute the friction force on $\mathbf{x}_i$. Our formulation simultaneously considers all the objects in contact with each cloth vertex, leading to more accurate friction forces compared to the sequential computation in most cloth simulators.

Suppose a cloth vertex is in contact with M triangles, we denote the normal contact forces and tangential contact forces as $\mathbf{f}_N$ and $\mathbf{f}_T$, where $\mathbf{f}_N, \mathbf{f}_T \in \mathbb{R}^{3M}$. $\mathbf{f}_N$ is computed either from the rigid body simulation or from the repulsive force in the cloth simulator, but $\mathbf{f}_T$ is an unknown vector. We formulate the following equations based on Coulomb's friction model:

$$(\mu_i \|\mathbf{f}_{N_i}\| - \|\mathbf{f}_{T_i}\|)(\|\mathbf{v}_i^r + h\frac{\sum_{j=1}^{M} \mathbf{f}_{T_j}}{m}\|) = 0, \tag{26}$$

$$\mu_i \|\mathbf{f}_{N_i}\| - \|\mathbf{f}_{T_i}\| \geq 0, \tag{27}$$

where $1 \leq i \leq M$. μ is the friction coefficient, $\mathbf{v}^r$ is the relative tangential velocity at the contact, h is the simulation time step, and m is the mass of a cloth vertex. The subscript i or j indicates the contact with the i-th (or j-th) triangle. The first constraint describes a complementarity condition; either the relative tangential velocity at the next time step is zero (static friction), or the contact force lies on the

boundary of the friction cone (dynamic friction). The second constraint enforces the contact force to lie in the friction cone.

Solving a nonlinear complementarity problem is very difficult and time consuming for a large number M. Fortunately, for cloth simulation, vertices collide with no more than two triangles, one on each side of the cloth. With $M = 2$, we can solve the problem efficiently by checking all possible cases. First, we assume that both contacts are static, leading to the following equations:

$$\|\mathbf{v}_1^r + h\frac{\mathbf{f}_{T_1} + \mathbf{f}_{T_2}}{m}\| = 0, \tag{28}$$

$$\|\mathbf{v}_2^r + h\frac{\mathbf{f}_{T_1} + \mathbf{f}_{T_2}}{m}\| = 0, \tag{29}$$

$$\mu_1\|\mathbf{f}_{N_1}\| - \|\mathbf{f}_{T_1}\| \geq 0, \tag{30}$$

$$\mu_2\|\mathbf{f}_{N_2}\| - \|\mathbf{f}_{T_2}\| \geq 0. \tag{31}$$

If solutions for $\mathbf{f}_{T_1}$ and $\mathbf{f}_{T_2}$ do not exist, we switch to the case where one contact is static and the other one is dynamic. We first choose the contact with larger $\mu_i\|\mathbf{f}_{N_i}\| - \frac{m\|\mathbf{v}_i^r\|}{h}$ to be dynamic. This heuristic is based on the static condition and the approximation of $\|\mathbf{f}_{T_i}\|$:

$$\mu_i\|\mathbf{f}_{N_i}\| - \|\mathbf{f}_{T_i}\| \approx \mu_i\|\mathbf{f}_{N_i}\| - \frac{m\|\mathbf{v}_i^r\|}{h}. \tag{32}$$

Suppose the first contact is set to be dynamic. We replace Equation 28 with $\mu_1\|\mathbf{f}_{N_1}\| - \|\mathbf{f}_{T_1}\| = 0$ and delete Equation 30. If we cannot find a solution for either contact being static, we relax both contacts and make them dynamic.

Once the friction force $\mathbf{f}_T$ is solved, we can update the velocity of the vertex i by:

$$\tilde{\mathbf{v}}_i^{n+1} = \bar{\mathbf{v}}_i^{n+1} + h\frac{\sum_{j=1}^{M}\mathbf{f}_{T_j}}{m}. \tag{33}$$

Note that the friction forces are computed implicitly considering the velocity of the vertex at the next time step. Directly using $\bar{\mathbf{v}}_i^{n+1}$ (that is the velocity without the effect of collision) to compute friction force leads to unstable simulation in our experiments.

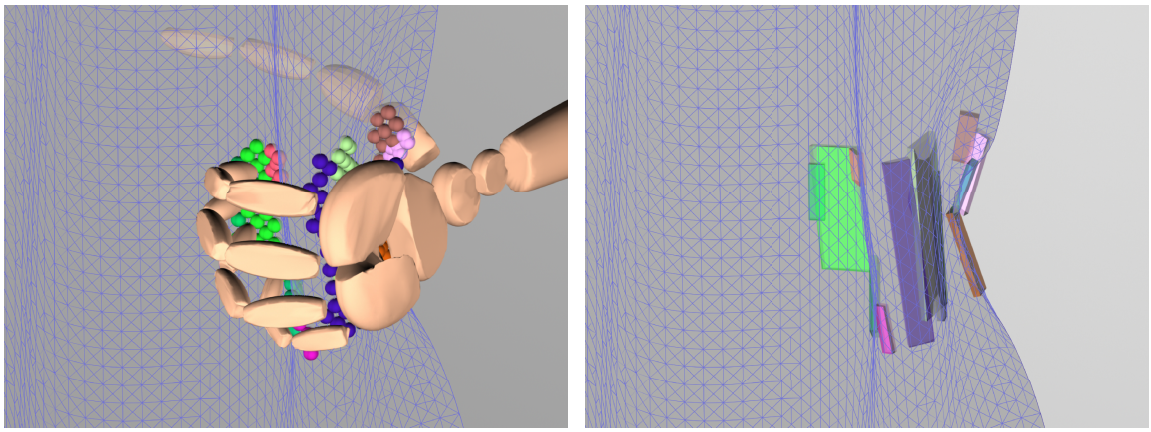


Figure 24: Illustration of rigid cloth patches. Left: Clustered vertices in the cloth triangle mesh. Right: Rigid cloth patches built from the clustered vertices.

5.5 Rigid Cloth Patch

Our algorithm approximates the state of cloth using a collection of thin rigid bodies, which we call rigid cloth patches. Each rigid cloth patch is a local representation of a small piece of cloth currently in contact with the rigid bodies. At each time step, we dynamically create rigid cloth patches to best fit the geometry of the cloth and assign them appropriate velocities. Once initialized, the rigid cloth patches participate in the rigid body simulation in the same way as other rigid bodies in the scene.

We group all the cloth vertices in $\mathcal{V}_r$ into rigid cloth patch clusters. A vertex i and a vertex j belong to the same cluster if they satisfy the following conditions: their distance is within a threshold in the world space (Equation 34), their normals are in the similar direction (Equation 35), and the displacement in the normal direction is sufficiently small (Equation 36):

$$\|\mathbf{x}_i - \mathbf{x}_j\| \leq \delta_w, \tag{34}$$

$$\|\mathbf{n}_i - \mathbf{n}_j\| \leq \delta_n, \tag{35}$$

$$\|(\mathbf{x}_i - \mathbf{x}_j)^{\mathbf{T}}\mathbf{n}_i\| + \|(\mathbf{x}_i - \mathbf{x}_j)^{\mathbf{T}}\mathbf{n}_j\| \leq \delta_d. \tag{36}$$

We set the threshold from Equation 34 to Equation 36 to be 40 mm, 20°, and 3 mm respectively.

For each cluster, we determine the orientation of the corresponding rigid cloth patch by fitting a plane to all the vertices in the cluster. The area of the rigid cloth patch is the minimum rectangle that covers all the vertices projected on the plane. The width of the rigid cloth patch is determined by the thickness of the cloth used in cloth simulator. In addition to geometry properties, we also need to assign the mass and the velocity to each rigid cloth patch. The mass is determined by the total mass of the cloth vertices in the cluster. For the velocity, we approximate the linear velocity of the rigid cloth patch using the average velocity of its corresponding cloth vertices. The angular velocity of the rigid cloth patch is approximated based on the total angular momentum of the corresponding cloth vertices with respect to the center of mass of the rigid cloth patch.

5.6 Hand Control

We designed a few simple hand controllers to perform grasping, pinching, and pressing the cloth with one or both hands. The controllers generate appropriate torques, $\boldsymbol{\tau}_{int}$ which are added to the equations of motion by the rigid body simulator:

$$\mathbf{M}(\mathbf{q}_h)\ddot{\mathbf{q}}_h + \mathbf{C}(\mathbf{q}_h, \dot{\mathbf{q}}_h) = \boldsymbol{\tau}_{ext} + \boldsymbol{\tau}_{int}, \tag{37}$$

where $\mathbf{q}_h$, a subset of $\mathbf{q}$, are the degrees of freedom of the hand model. In Equation 37, $\mathbf{M}$ and $\mathbf{C}$ are the mass matrix and the Coriolis and centrifugal forces expressed in generalized coordinates, $\boldsymbol{\tau}_{ext}$ are the sum of all external forces, including the collision forces with rigid cloth patches.

All the controllers are developed based on proportional-derivative (PD) actuators at joints to track a desired hand pose created either manually or by inverse kinematics. For some manipulations, such as pressing a cloth on the table, we use Jacobian transpose control scheme to compute $\boldsymbol{\tau}_{int}$ such that the desired Cartesian force can be generated at the end-effector:

$$\boldsymbol{\tau}_{int} = \mathbf{J}_p^T \mathbf{F}. \tag{38}$$

Table 1: Computational performance breakdown: ARCSim (cloth), DART (rigid), and the Interface (our method). Time: the average of total simulation time (in second) per simulation time step. #Triangle: the number of triangles in the input mesh.

	ARCSim	DART	Interface	Time	#Triangle
Dressing	85 %	12 %	3 %	8.52s	8441
Curtain	97 %	2 %	1 %	5.30s	16384
Moving	98 %	1.7 %	0.3%	5.33s	8194

Here $\mathbf{J}_p$ is the Jacobian matrix at a point p on the hand and $\mathbf{F}$ is the desired Cartesian force.

5.7 Results

We apply our method to a variety of cloth manipulation scenarios shown in Figure 25. Our hand model is designed based on an anthropomorphic hand structure with 33 degrees of freedom. The breakdown of computation time and the number of triangles in the input mesh is shown in Table 1. On average, the computation due to our coupling interface is about 1% of the total computation time. The simulation time step is 5 ms. We set the repulsion thickness in ARCSim to 5 mm and the damping coefficient to 0.005.

5.7.1 Dexterous Manipulation Tasks

Pinching, moving, and folding cloth. We use the thumb and the index finger to pinch the cloth firmly and move it around. The fingers are controlled by using PD controllers to track a pose. External forces and torques are applied at the wrist to translate and rotate the hand. We also use two hands to grab the corners of a piece of cloth and fold it in half (Figure 25 (a)). These examples demonstrate that our method is able to generate appropriate friction to grasp the cloth without slipping.

Putting on a sleeve. Dressing requires some of the most intricate dexterous manipulation skills. To demonstrate the possibility of applying our method to this interesting problem, we design a simple controller that uses one hand to put a sleeve

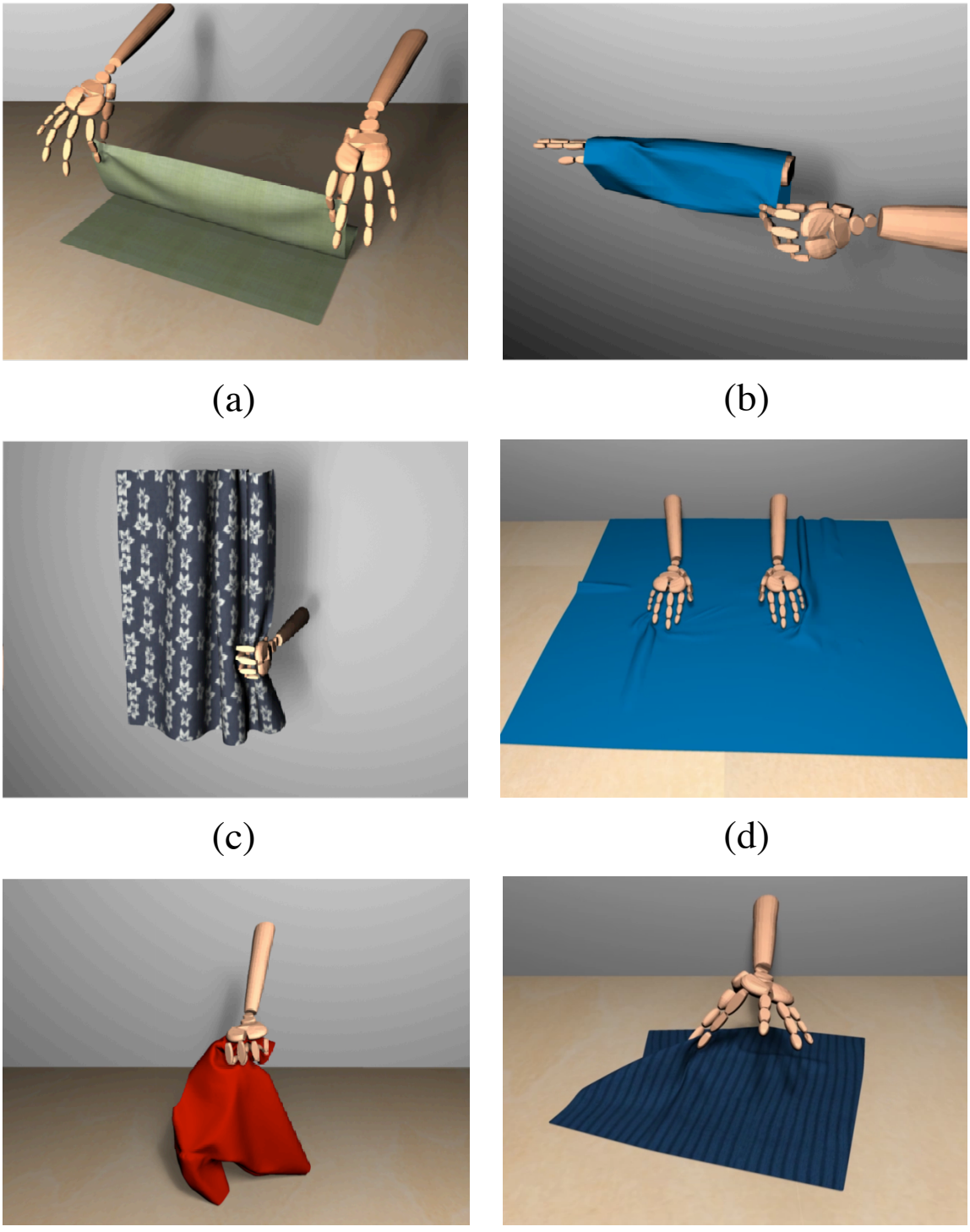


Figure 25: Dexterous manipulation of cloth.

on the other arm. The hand grabs the sleeve with the thumb, index finger, and middle finger (Figure 25 (b)). This example involves a large number of contact points between the cloth and the finger, and between the cloth and the forearm.

Pulling a curtain. Pulling a shower curtain presents a scenario of interaction between cloth and both the palm and fingers. We control the entire hand to grab the curtain and slide it sideways (Figure 25 (c)). This example illustrates the importance of rigid cloth patch simulation because the collision state between the hand and the cloth is complicated and rapidly changing over time.

Removing and generating wrinkles. By commanding two hands to press against a piece of cloth on the table, we can generate and remove wrinkles in the cloth. We use Jacobian transpose controllers to generate desired forces from the bottom of the palm. In the first example, the cloth presents asymmetric wrinkles in its initial configuration. When the hands slide across the cloth, the wrinkles are removed (Figure 25 (d)). Similarly, we show that the wrinkles could be generated by hand motion on a cloth that was initially smooth and flat.

Picking up cloth. We demonstrate two examples of picking up the cloth from different initial configurations. The first example attempts to pick up a t-shirt in an arbitrary initial configuration. We use the entire hand to create an envelope grasp when it reaches the cloth (Figure 25 (e)). In the second example, the hand picks up the cloth lying flat on the table. A common strategy to pick up cloth in this situation is to first use fingers to generate opposing friction forces to create a bunch, and then grab the bunch to lift the cloth. Our method successfully simulates this scenario due to our friction force computation (Figure 25 (f)).

5.7.2 Evaluations

Controlling contact forces. An important control strategy in dexterous manipulation is to control contact forces applied on the object. One benefit of our algorithm is

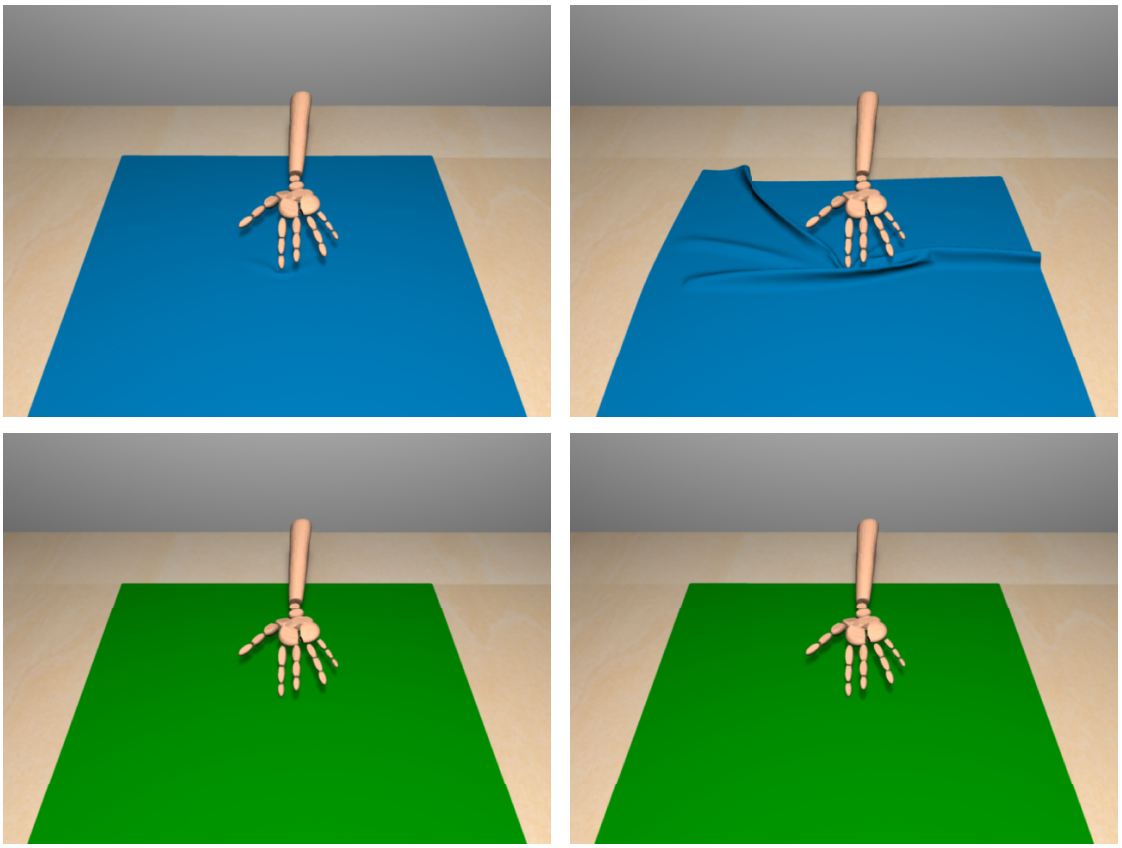


Figure 26: Controlling contact forces. Top row: Simulated results with our contact force computation in the situations of small (left) and large (right) pushing forces. Bottom row: Simulated results with ARCSim contact force computation in the situations of small (left) and large (right) pushing forces.

that the simulated cloth reacts differently when the hand applies different amounts of force to it. In Figure 26, we compare the simulation results with different push forces. When the finger exerts a small pressing force on the cloth, the finger can smoothly slide on the cloth. When the force is large, wrinkles are generated due to the large friction. However, when the same scenarios are simulated using ARCSim, shown in the bottom row, the cloth motion shows no difference between two forces. This is because the contact force is computed based only on the relative position and velocity between the hand and the cloth. In this case, no matter how large the contact force between the hand and the cloth is, the relative position and velocity remain the same.

Manipulating multiple layers of cloth. Many common manipulation scenarios

Figure 27: Collision with multiple cloth. Left: Simulated results of holding three pieces of cloth with (left) and without (right) rigid cloth patches.

involve grasping multiple layers of cloth. Figure 27 demonstrates a scenario with three pieces of cloth held together by the hand. ARCSim typically has a difficult time handling such a scenario. When the fingers are far apart, the cloth cannot be held, and when the fingers are too close, the complex collision issues cannot be resolved. We show that, due to the accurate approximation of cloth using rigid cloth patch simulation, our algorithm is able to hold three pieces of cloth together without any collision issues.

5.8 Limitations

Although we make the decision not to implement accurate two-way coupling for the reason of having a simple implementation and efficient computation, we recognize that some examples of dexterous manipulation require that the rigid bodies react to the dynamics of the cloth accurately. In particular, when the manipulator uses the cloth in hand to interact with other rigid objects, the forces from the cloth to the rigid objects might not be negligible. For example, putting on a sock probably does not require two-way coupling between the hands and the sock, but the forces exchange between the sock and the foot cannot be ignored.

In the cloth simulator, the cloth refinement steps, that is remeshing, inelastic

projection and strain limit correction sometimes interfere with the contact force computed by our algorithm. The issue is more evident for static contact points. Without these refinement steps, the contact forces computed by our algorithm are able to enforce a static contact. However, when we activate the refinement procedures, the static contacts can be moved to different locations at the end of the simulation step. Therefore, we use position constraints only during the refinement steps to ensure that these vertices are not altered as a side effect of the refinement steps. Note that the position constraints do not add any forces or change the dynamic state of the system. If we skip the refinement steps, the position constraints will have no effect on the algorithm.

There are some limitations of using rigid cloth patches. First, the flexibility of cloth can be limited by the rigid cloth patch approximation. In our implementation, however, the rigid cloth patches are reconstructed at every time step and can be as small as the triangles on the cloth. By using small-scale rigid cloth patches and frequent reconstruction, we find the rigid cloth patch approximation to be sufficient for the hand manipulation tasks we demonstrated in this work. Second, the rigid cloth patches are not sufficiently accurate to provide the friction forces between the rigid body and the cloth. This is the reason why we only use rigid cloth patches to approximate the geometric state of the cloth and computed friction forces by solving the nonlinear complementarity problem.

5.9 Conclusion

In this chapter, we introduce a simulation technique that couples cloth and rigid body simulations to synthesize dexterous manipulation of cloth. Our technique is built upon existing cloth and rigid body simulators. We focus on two main issues in current cloth simulators. First, the contact force computation does not take into account the dynamic information of the rigid body. Second, rigid body motion is unaware

of the state of cloth, which frequently causes unsolvable collision situations. We develop a light-weight interface through which the rigid body and the cloth simulators communicate on a demand-driven manner to resolve these two issues. We demonstrate a set of basic manipulation skills, such as gripping, pinching, and pressing.

This work is intended to be a simple and practical solution to couple existing rigid body simulators and cloth simulators for dexterous manipulation of deformable objects. We believe that it is an important missing piece, which will enable many interesting future research problems in control and simulation. For example, developing controllers for tying shoelaces or folding laundry can benefit robotic manipulators in numerous ways. We also envision that this technique can be integrated with full-body character simulation and control. Developing a virtual character capable of dressing itself may also be a fruitful future research direction.

This chapter solves the problem of coupling hands and cloth in simulation. In the next chapter, we will address another challenge focusing on the control of cloth motion using hands.

CHAPTER VI

DEXTEROUS MANIPULATION OF CLOTH

In this chapter, we introduce a new control technique to synthesize dexterous manipulation of cloth based on the simulation technique we present in Chapter 5. Given a simple description of the desired cloth motion, our algorithm computes appropriate joint torques for a physically simulated hand, such that, via contact forces, the result of cloth simulation follows the desired motion. Instead of optimizing the hand control forces directly, we formulate an optimization problem that solves for commanding force, which has more direct impact on the dynamic state of the hand and that of the cloth. The solution of the optimization provides commanding forces that achieve desired cloth motion described by the user, while respecting the kinematic constraints of the hands. These commanding forces are then used to guide the joint torques of the hand. To balance between the effectiveness of control and computational costs, we formulate a model-predictive-control problem as a quadratic program at each time step. We demonstrate our technique on a set of cloth manipulation tasks in daily activities, including folding laundry, wringing a towel, and putting on a scarf.

6.1 Introduction

Dexterous manipulation of cloth is a unique human skill essential to many activities of daily living spanning from dressing to grooming to folding laundry. Synthesizing these activities automatically for computer animation requires realistic depiction of human dexterity and cloth simulation. An experienced artist with the aid of a state-of-the-art cloth simulator might be able to animate simple manipulation task through trial-and-error, but the complex interaction of dexterous manipulation and cloth motion demands more sophisticated automatic solution.

Dexterous manipulation and user-control of cloth simulation are two active computer animation research areas that have rarely crossed path in the past. Researchers have demonstrated control algorithms on simulated hands to achieve impressive dexterous tasks, but the scope has been limited to manipulating rigid objects. Deformable objects, such as cloth, present new challenges because the hand has to manipulate a much larger number of uncontrollable degrees of freedom of cloth. Furthermore, the complex relation of the cloth state and contact forces increases the difficulty of dexterous controller design. On the other hand, a separate research community has developed various algorithms to control physically simulated cloth according to user-specifications. These techniques aim to produce the cloth motion alone without consideration of human hands, thereby unphysical force is allowed to apply anywhere on the cloth. Directly applying existing techniques from either area is insufficient to address the new challenges of dexterous manipulation of cloth.

In this chapter, we propose a new technique to consolidate the constraints imposed by hand manipulation and control of cloth simulation. In our framework, the artist provides desired trajectories of a small set of features on the cloth (for example vertices at corners of the cloth) and expects the resulting cloth motion to follow the input feature trajectories through the contact forces provided by the virtual hands. Our algorithm formulates a reduced optimization problem that solves for the commanding forces from the hand to the cloth, a pivotal physical quantity that couples the state of the hand and that of the cloth. The optimization aims to find the optimal commanding forces that achieve desired cloth motion described by the user, while respecting constraints from contact dynamics and hand kinematics. To balance between the effectiveness of control and computational costs, we formulate a model-predictive-control (MPC) problem as a quadratic program (QP) at each time step.

We demonstrate our method on a variety of cloth manipulation tasks such as

Figure 28: Given a simple description of the desired cloth motion, our algorithm computes appropriate joint torques for a physically simulated hand, such that, via contact forces, the result of cloth simulation follows the desired motion.

folding laundry, wringing a towel, or putting on a scarf (Figure 28). To validate the cloth motion planning algorithm quantitatively, we randomly generate a set of feature trajectories in a defined space and measure the accuracy of the resultant cloth motion. The evaluation shows that our cloth motion planning can accurately achieve the user-specifications on the cloth and can be successfully executed by the simulated hands.

6.2 Related Work

In Chapter 5, we discuss the importance of accurate two-way coupling between cloth and rigid body to realize dexterous hand manipulation of cloth. A simple task of pinching a piece of cloth between two fingers could fail without accurate contact force and successful collision handling. Likewise, simulated hands will not be able to create budges or wrinkles on the cloth without appropriate friction forces. Researchers

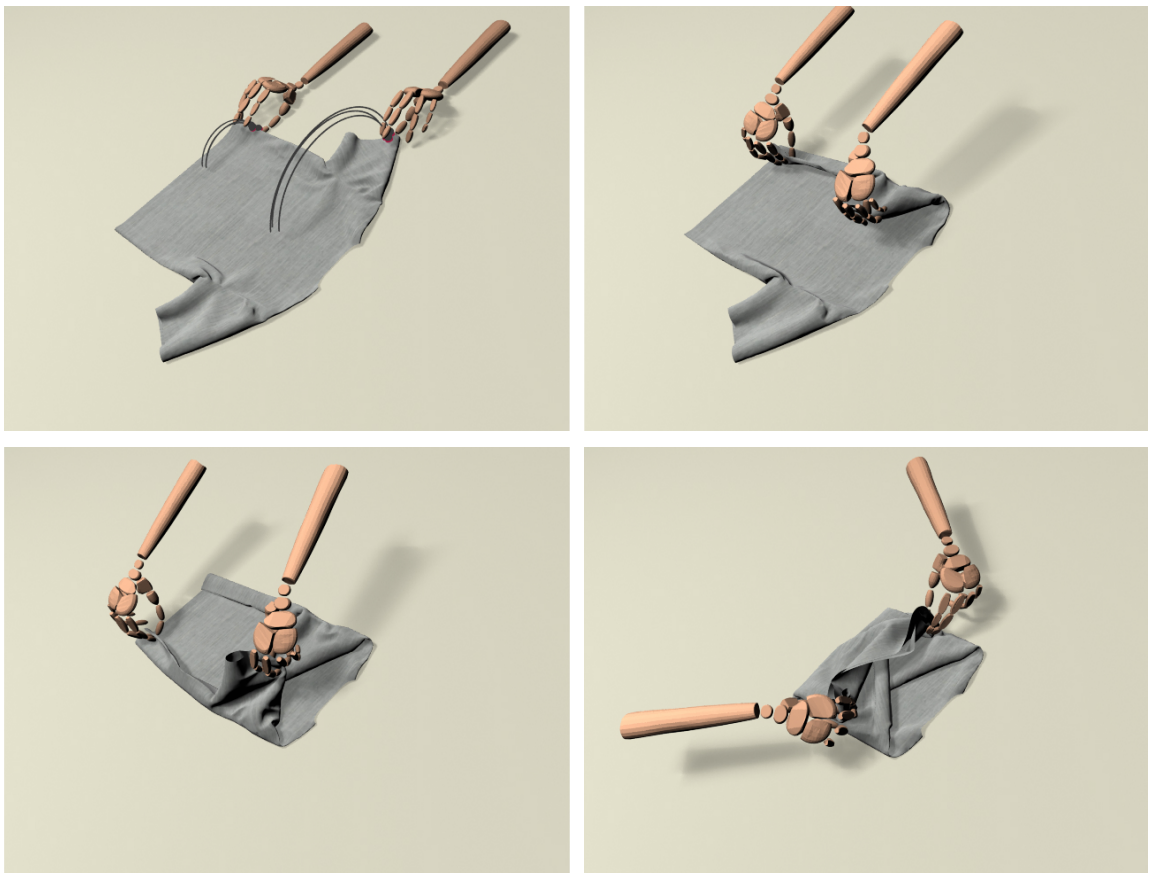


Figure 29: Fold a T-shirt.

have proposed various methods to handle two-way coupling between rigid bodies and deformable bodies [57, 116, 115, 101, 89, 12]. In Chapter 5, we present a method to couple cloth and rigid body simulation using the existing simulators as black box. Our method allows rigid bodies to impart friction forces on cloth and avoids unsolvable collision situations between the rigid bodies and the cloth. In this chapter, we use this method for handling two-way coupling because we desire to use existing rigid body and cloth simulators. However, in Chapter 5, we only focus on simulation algorithms and only demonstrate a few manually-designed grasp controllers insufficient for more complex manipulation tasks of cloth. In this chapter, our goal is to provide more general computational tools suitable for a wide set of manipulation tasks in addition to grasping.

Controlling simulated cloth motion for computer animation and special effect applications has been investigated extensively [133, 18, 17, 49]. Many approaches formulated a large spacetime optimization and solved it efficiently using different methods, such as adjoint method [133] or model reduction [17, 49]. Bergou et al. used low-resolution cloth simulation as preview and tracked this motion with high-resolution simulation to enhance visual details and fidelity [18]. These methods can generate desired cloth motion specified by the user, but they allow for unphysical "control force" applied on the cloth, as if the cloth is an actively actuated object. In this chapter, we will present a cloth motion planning algorithm that is similar in that it creates desired cloth motion according to user specifications, but the control force applied on the cloth must be caused by the movement of hands in contact with the cloth.

Though still in its infancy, a few methods have demonstrated the potential and the importance of cloth manipulation in character animation [50, 88, 128, 129, 29]. Ho and Komura introduced a technique to interact with deformable bodies using topology coordinates [50]. Wang et al. showed a virtual human putting on a sock and a pair of shorts by using electric flux for path planning [129]. Most recently, Clegg et al. proposed a method for animating human dressing [29]. However, the interaction between hand and cloth is achieved simply through position constraint. In this chapter, we are interested in animating manipulation of cloth with dexterous hand through simulated contact and friction.

6.3 Overview

Most cloth manipulation tasks can be broken down to a sequence of *contact phases*. In each contact phase, a *grasp/regrasp* action first establishes contact between the hand and the cloth followed by a *manipulation* action that brings the cloth to a desired state. For example, the folding task shown in Figure 29 requires three contact phases, each of which involves grasping a corner of the cloth followed by moving the corner

to the desired location.

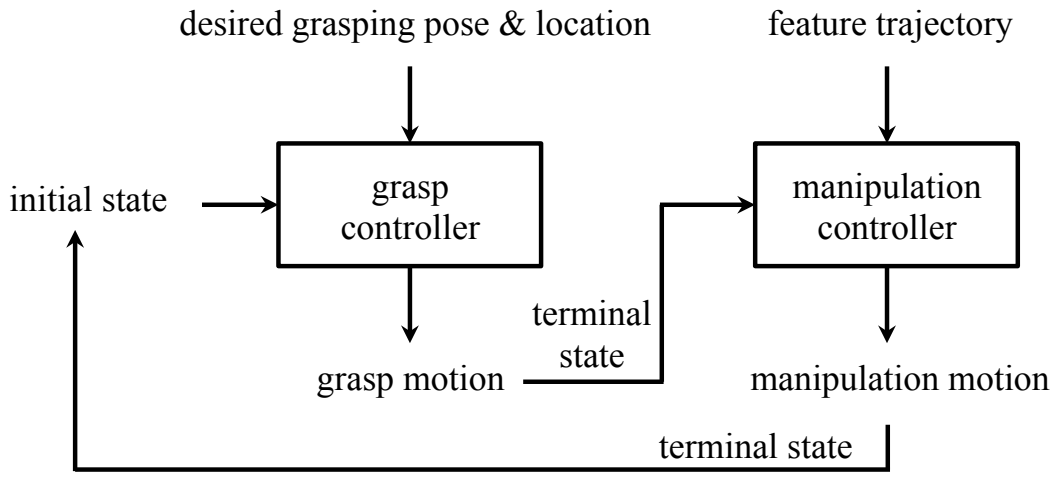


Figure 30: Our algorithm consists of two components: a grasp controller and a manipulation controller.

Based on this observation, our algorithm (Figure 30) consists of two components: a grasp controller (Section 6.5) and a manipulation controller (Section 6.4). The grasp controller takes as input the desired grasping pose and the desired grasping location on the cloth and outputs a sequence of hand and cloth motion which terminates when the cloth is firmly gripped by the hand in the desired pose at the intended location. The terminal state of the grasp motion becomes the initial state of the manipulation controller. Starting from this initial state, the manipulation controller takes as input the desired motion of cloth described by a few feature trajectories and simulates the motion of hand manipulating the cloth. A feature can be any linear function of the vertices on the cloth. The final state of the manipulation motion becomes the initial state of the next contact phase.

6.4 Manipulation Controller

The manipulation controller aims to achieve the desired motion of the cloth through the actions of the hand. At each time step, the manipulation controller consists of three consecutive steps: cloth motion planning, rigidity rectification, and hand

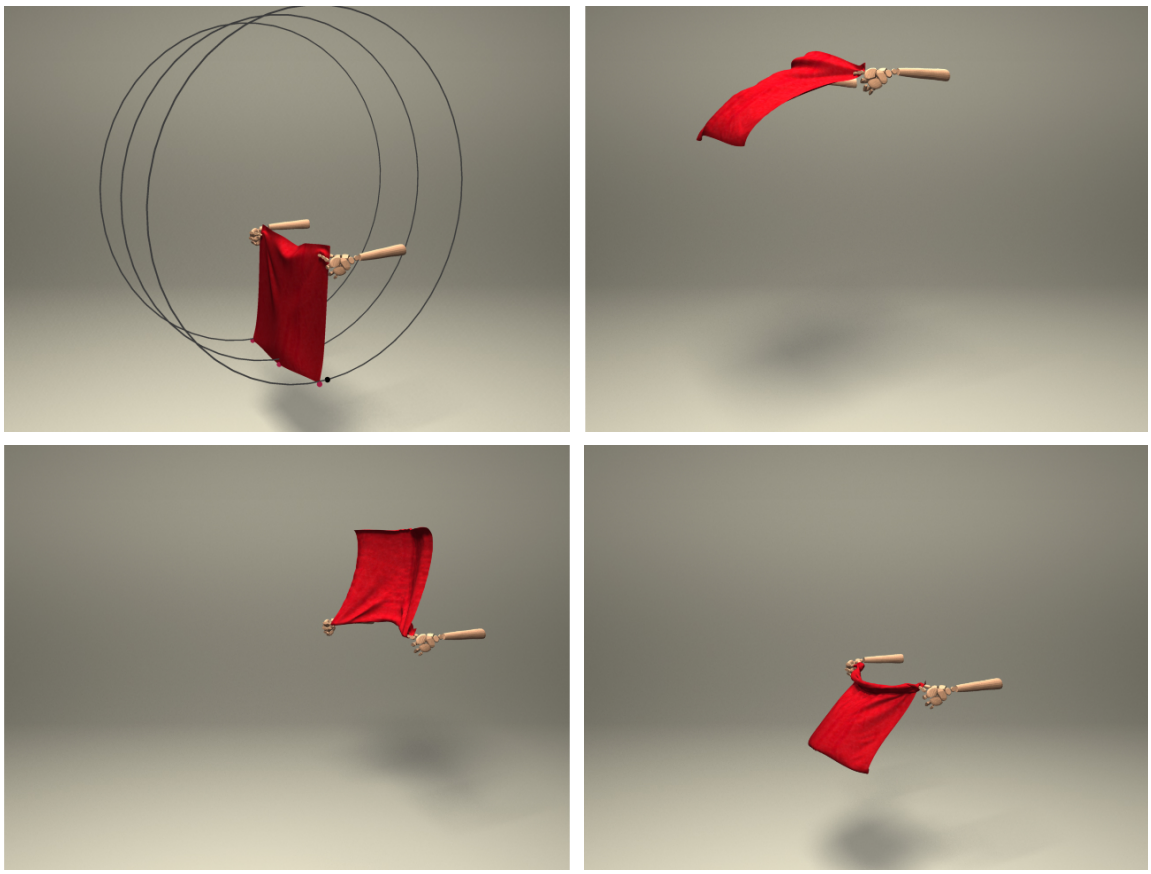


Figure 31: A dynamic secondary motion can be controlled by our algorithm.

control. Cloth motion planning, formulated as a model-predictive-control problem, solves for required commanding forces from the hand such that the cloth tracks the user-specified feature trajectories closely. Rigidity rectification makes sure that the movement of contact points from the plan is achievable by the grasping hand. Finally, the rectified contact points are used to guide the actuation of the grasping hand.

6.4.1 Cloth Motion Planning

Directly solving for hand actuation to manipulate the cloth is very challenging due to complex dynamic coupling between the state of the hand $(\mathbf{q}, \dot{\mathbf{q}})$ and the state of the cloth $(\mathbf{u}, \dot{\mathbf{u}})$. Instead, our computation focuses on solving commanding forces, $\mathbf{f}_c \in \mathbb{R}^{3n}$, where n is the number of vertices of the cloth. The goal of the motion planning is to produce $\mathbf{f}_c$ that makes the cloth track the user-specified feature trajectories, while

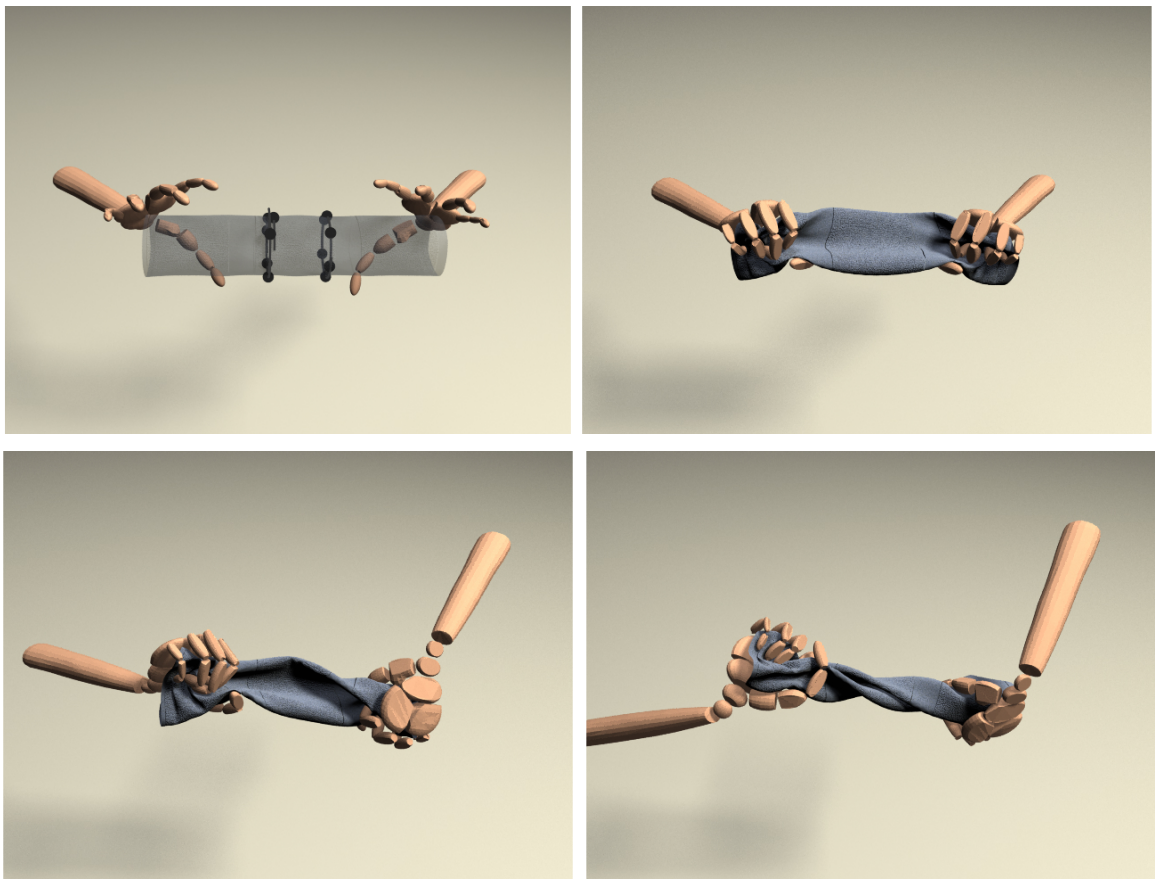


Figure 32: Wring a towel.

being achievable by the hands through the contact points.

At each time step i, we formulate a short-horizon optimization to solve for a small window of commanding forces, $\mathbf{f}_c^{i+1}, \cdots, \mathbf{f}_c^{i+K}$, from frame i. We denote the time difference from the current frame by index k and the window size by K. After solving all the commanding forces in the current window, we use the state of the cloth integrated with the first commanding force $\mathbf{f}_c^{i+1}$ and other forces in the system to guide the control of the hand (Section 6.4.3). With the coupled simulation of hand and cloth, we obtain the next state of cloth, $(\mathbf{u}^{i+1}, \dot{\mathbf{u}}^{i+1})$. At the next time step, the optimization window slides one step forward and a new plan for the next K frames (Table 2 lists values of K in our implementation) of commanding forces is optimized. The remaining of Section 6.4.1 describes how this short-horizon optimization can be

formulated as a quadratic program and solved efficiently.

Physical feasibility. While we wish to avoid solving the full state of the hand in the optimization, we still need to make sure that the grasping hand is capable of producing $\mathbf{f}_c$. As such, we assume that the commanding forces can only apply at static contact points and attempt to maintain static contact during manipulation. This assumption results in a conservative control strategy that does not explicitly exploit slipping or rolling strategies, but it is still sufficient to achieve a wide range of cloth manipulation tasks in daily life as shown in the result section.

The algorithm first selects the vertices that are in static contact with finger in the previous time step. For each finger o at every time step i, our algorithm identifies a selected set of vertices on the cloth, $\hat{\mathbf{u}}_j \in \mathcal{P}_o^i$, that are in static contact with the finger at frame $i-1$. The finger o can intentionally apply commanding forces to manipulate the cloth only through $\hat{\mathbf{u}}_j$. The control strategy obeys the Coulomb friction cone condition for static contact by expressing $\mathbf{f}_c$ as:

$$\mathbf{f}_c = [(\mathbf{B}_1\mathbf{d}_1)^T(\mathbf{B}_2\mathbf{d}_2)^T \cdots (\mathbf{B}_n\mathbf{d}_n)^T]^T. \tag{39}$$

For those cloth vertices not in the union of all $\mathcal{P}_o^i$'s, we directly set $\mathbf{Bd} = \mathbf{0}$. For the rest of cloth vertices, the commanding force is expressed as the bases of the approximated friction cone $\mathbf{B}$ multiplied by coefficients $\mathbf{d}$. The cloth vertex may have multiple fingers in contact with it (for example one finger from each side of cloth). Therefore, $\mathbf{B} \in \mathbb{R}^{3\times 4m}$ is the combined friction cone, and $\mathbf{d} \in \mathbb{R}^{4m}$ is the combined coefficients, where m is the number of fingers in contact. $\mathbf{B}$ is determined by the normal direction and the friction coefficient. The friction cone condition requires

$$\mathbf{d} \geq \mathbf{0}. \tag{40}$$

The friction cone coefficients, $\mathbf{D} = [\mathbf{d}_1^T \cdots \mathbf{d}_n^T]^T$, will be the variables in the optimization described below.

We also wish that the vertices in $\mathcal{P}_o^i$ move under the same rigid transformation as they are gripped by the same finger. However, the rigid transformation will result in nonlinear constraints unsuitable for a quadratic program. As such, we rely on rigidity rectification (Section 6.4.2) to enforce rigid motion and relax the rigid transformation requirement to linear transformation in the QP optimization:

$$(\hat{\mathbf{u}}_j^{i+k} - \mathbf{c}^{i+k}) = \mathbf{L}^{i+k}(\hat{\mathbf{u}}_j^i - \mathbf{c}^i). \tag{41}$$

Here $\mathbf{c} \in \mathbb{R}^3$ is the center of mass of the vertices $\hat{\mathbf{u}}_j \in \mathcal{P}_o$. $\mathbf{L}^{i+k} \in \mathbb{R}^{3\times3}$ is the linear transformation of $\hat{\mathbf{u}}_j$ relative to $\mathbf{c}$ from frame i to frame $i+k$. We solve for $\mathbf{L}$ together with $\mathbf{D}$ in the optimization.

The commanding forces also need to obey the laws of physics governing the cloth motion:

$$\mathbf{M}\Delta\dot{\mathbf{u}} = h\mathbf{G}(\mathbf{u}) + h\mathbf{f}_e + h\mathbf{f}_c, \tag{42}$$

where $\mathbf{M} \in \mathbb{R}^{3n\times3n}$ is the mass matrix of cloth mesh, $\mathbf{u} \in \mathbb{R}^{3n}$ is the position vector of all the cloth vertices, $\mathbf{G}$ includes bending and stretching forces of the cloth mesh, $\mathbf{f}_e$ is other external forces (for example gravity or incidental contact forces with other objects in the scene), and h is the time step size ($h = 0.02$s in our implementation). Note that $\mathbf{G}$ is a nonlinear function of $\mathbf{u}$, but we could linearize $\mathbf{G}$ about the current state $\mathbf{u}^i$ to meet the requirements of quadratic programming:

$$\mathbf{M}\Delta\dot{\mathbf{u}}^{i+k} = h\mathbf{G}(\mathbf{u}^i) + h\frac{\partial\mathbf{G}}{\partial\mathbf{u}}|_{\mathbf{u}^i}(\mathbf{u}^{i+k} - \mathbf{u}^i) + h\mathbf{f}_e^{i+k} + h\mathbf{f}_c^{i+k}. \tag{43}$$

Because $\Delta\dot{\mathbf{u}}$ can be expressed as a linear function of $\mathbf{f}_c$ via Equation 43, the cloth positions in the time window are also linear functions of $\mathbf{f}_c$ via implicit Euler time integration:

$$\mathbf{u}^{i+k} = \mathbf{u}^i + kh\dot{\mathbf{u}}^i + h\sum_{j=0}^{k-1}(k-j)\Delta\dot{\mathbf{u}}^{i+1+j}. \tag{44}$$

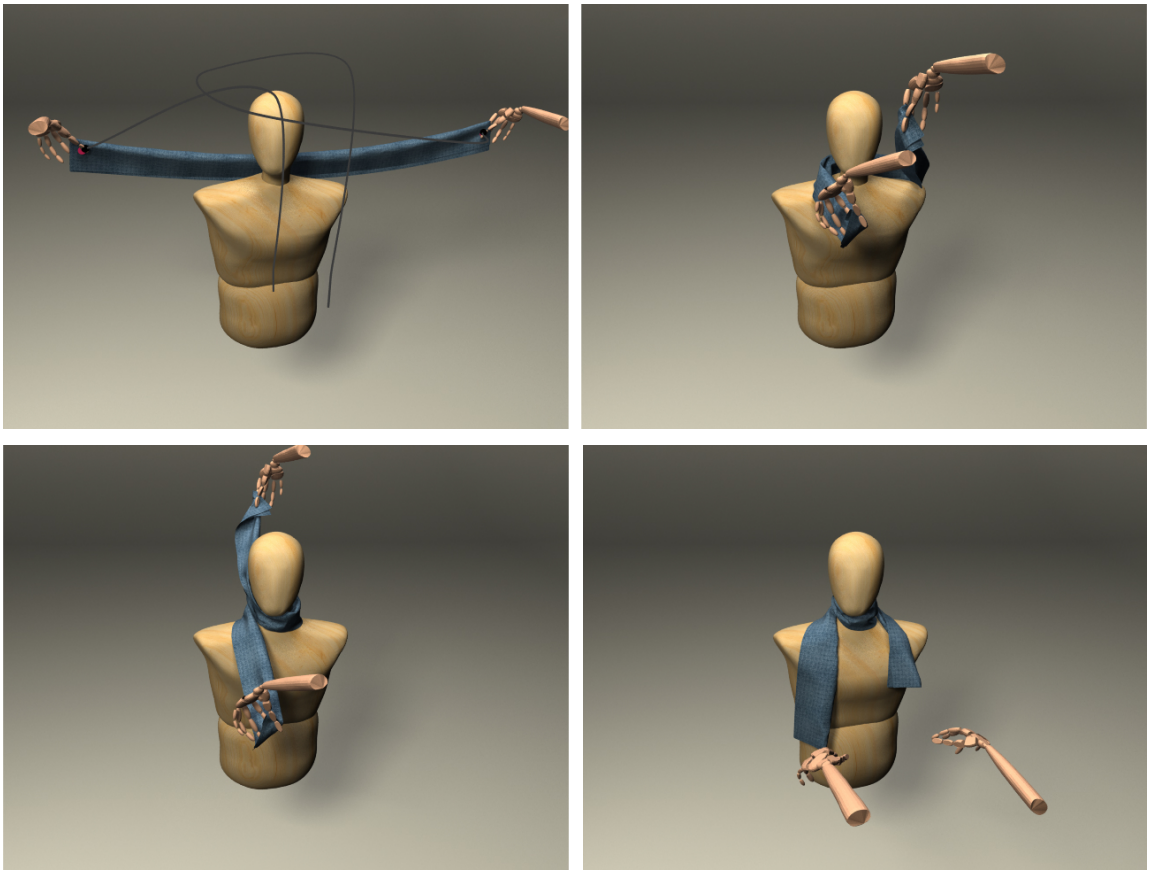


Figure 33: Put on a scarf.

Objective function. In addition to physical feasibility, the optimization considers three objectives. The first objective is to closely follow the desired cloth motion described by the user-defined feature trajectories, $\bar{\phi}$, represented as b-splines:

$$E_f = \sum_{k=1}^{K} \|\phi(\mathbf{u}^{i+k}) - \bar{\phi}^{i+k}\|^2, \tag{45}$$

where $\phi(\mathbf{u}^{i+k})$ evaluates the features using the cloth configuration at frame $i+k$.

The second objective serves as a regularization term to minimize the magnitude of commanding force, preventing unnaturally large commanding forces generated by the human hand:

$$E_m = \sum_{k=1}^{K} \|\mathbf{f}_c^{i+k}\|^2. \tag{46}$$

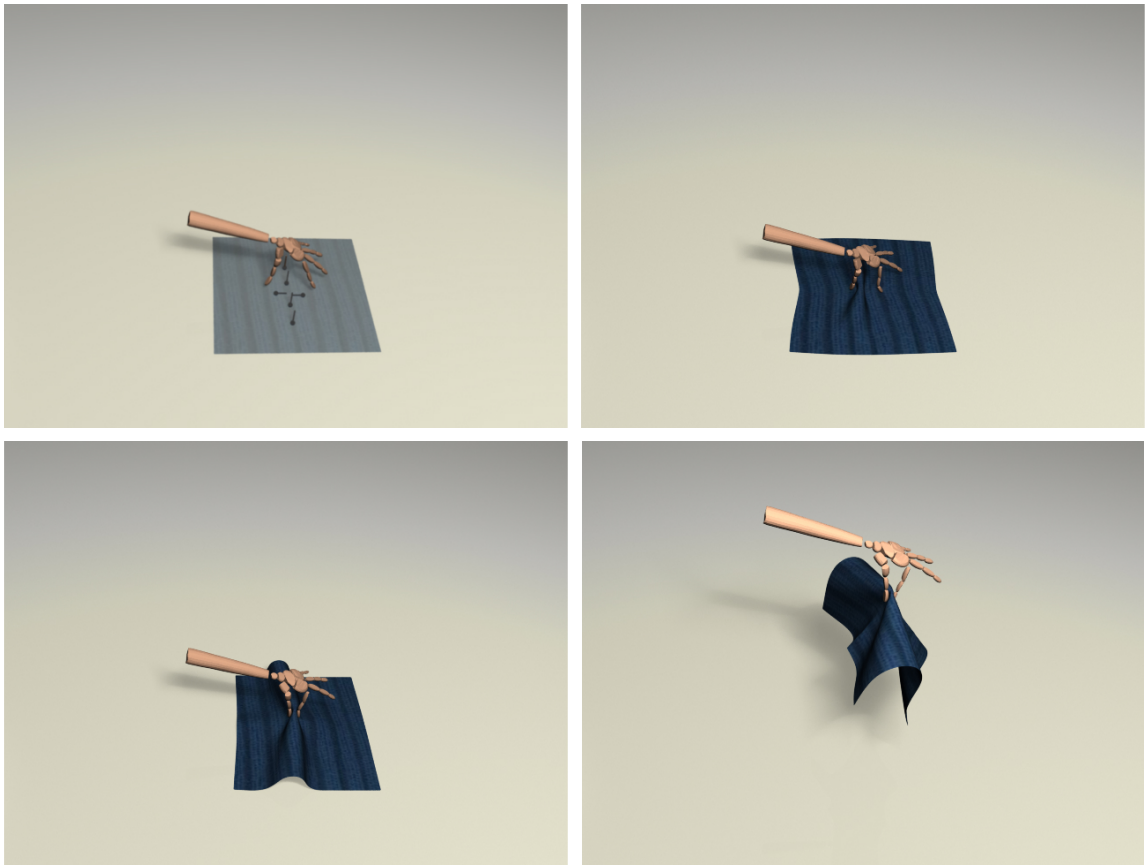


Figure 34: Lift from a flat surface.

We also wish to encourage the contact points in each $\mathcal{P}_o$ to move smoothly:

$$E_s = \sum_{k=1}^{K} \|\dot{\mathbf{c}}^{i+k} - \dot{\mathbf{c}}^{i+k-1}\|^2 + \sum_{k=2}^{K} \|\mathbf{L}^{i+k} - \mathbf{L}^{i+k-1}\|^2 + \|\mathbf{L}^{i+1} - \mathbf{I}\|^2, \tag{47}$$

where $\mathbf{I}$ is the identity matrix.

Formulating quadratic program. Finally, we formulate a quadratic program at each time step i as:

$$\begin{aligned} \underset{\mathbf{D}^{i+k}, \mathbf{L}^{i+k}, 1 \le k \le K}{\text{argmin}} \quad & \omega_f E_f + \omega_m E_m + \omega_s E_s \\ \text{subject to} \quad & \text{Equation 40, 41.} \end{aligned} \tag{48}$$

Once we solve $\mathbf{D}$, we can recover $\mathbf{f}_c$ from Equation 39. Equation 43 is implicitly enforced as we express $\mathbf{u}^{i+k}$ as a linear function of $\mathbf{f}_c$ in the optimization. ω_f, ω_m,

and ω_s are weights for each objective term, and are set to 40, 0.01, 10 respectively in our implementation.

Once we solve the commanding forces in the current window, we could simply apply the first commanding force $\mathbf{f}_c^{i+1}$ on the cloth to advance the cloth to the next state $(\mathbf{u}^{i+1}, \dot{\mathbf{u}}^{i+1})$, and repeat this process to simulate a cloth sequence that achieves desired motion if our goal were to create cloth motion without consideration of the hands. In our case, however, instead of directly applying $\mathbf{f}_c$ to the cloth, we need to compute the appropriate actuation of the hands to move the cloth in the same way as $\mathbf{f}_c$ does via physical simulation with realistic contact forces.

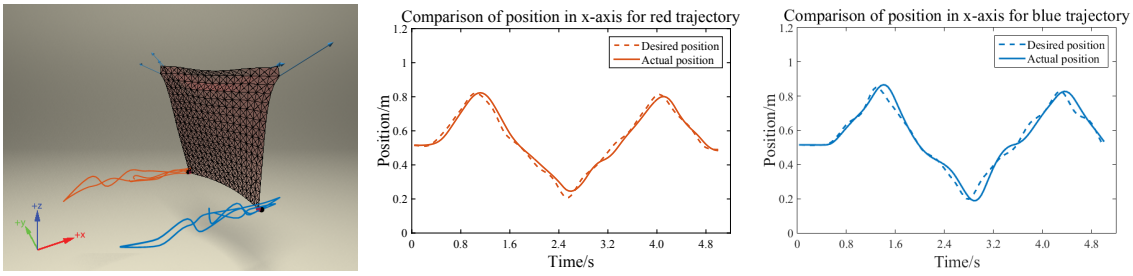
6.4.2 Rigidity Rectification

Before we solve for the hand actuation, we need to make sure that the commanding forces from cloth motion planning are feasible for the hands. Equation 39, 40 and 41 improve the feasibility to some extent, but the relaxation of rigid transformation for formulating a QP can potentially cause undesired contact movements for the hands, that is the contact vertices $\hat{\mathbf{u}}_j \in \mathcal{P}_o$ on the same finger exhibit non-rigid motion impossible for the finger o to follow. The rigidity rectification step applies the matching algorithm by Horn et al. [51] to factor out the rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ from $\mathbf{L}^{i+1}$ solved in Equation 48 and uses it to rectify each $\hat{\mathbf{u}}_j^{i+1} \in \mathcal{P}_o^{i+1}$:

$$\hat{\mathbf{u}}_j^{i+1} = \mathbf{R}(\hat{\mathbf{u}}_j^i - \mathbf{c}^i) + \mathbf{c}^{i+1}. \tag{49}$$

6.4.3 Hand Control

With the rectified contact vertices $\hat{\mathbf{u}}^{i+1}$, we can now compute the actuation of the hands such that the corresponding contact points on the hands follow $\hat{\mathbf{u}}^{i+1}$. We first apply inverse kinematics to compute a desired hand pose $\bar{\mathbf{q}} \in \mathbb{R}^m$ to match $\hat{\mathbf{u}}^{i+1}$, where m denotes the number of degrees of freedom of the hand. Next, we solve the internal force $\boldsymbol{\tau}_{int}$ at each actuated degree of freedom on the hands using stable PD



(a) (b) (c)

Figure 35: Evaluation of cloth motion planning. (a) We define two feature trajectories on the lower corners of a handkerchief while using two upper corners as contact points to provide control. The optimal commanding forces are shown as blue arrows. (b,c) We plot the desired position (dashed line) and the actual position (solid line) in x-axis of the two bottom corners.

(SPD) formulation [124]:

$$\boldsymbol{\tau}_{int} = -\mathbf{K}_p(\mathbf{q} + h\dot{\mathbf{q}} - \bar{\mathbf{q}}) - \mathbf{K}_d(\dot{\mathbf{q}} + h\ddot{\mathbf{q}}), \tag{50}$$

where $\mathbf{K}_p \in \mathbb{R}^{m\times m}$ and $\mathbf{K}_d \in \mathbb{R}^{m\times m}$ are diagonal matrices whose diagonal elements are set to 800 and 100 respectively. The acceleration of hand $\ddot{\mathbf{q}}$ can be evaluated via equations of motion:

$$\ddot{\mathbf{q}} = (\mathbf{M}_q + h\mathbf{K}_d)^{-1}(-\mathbf{C}_q - \mathbf{K}_p(\mathbf{q} + h\dot{\mathbf{q}} - \bar{\mathbf{q}}) - \mathbf{K}_d\dot{\mathbf{q}} + \boldsymbol{\tau}_{ext}), \tag{51}$$

where $\mathbf{M}_q$ and $\mathbf{C}_q$ denote the mass matrix, Coriolis and centrifugal force in generalized coordinates. $\boldsymbol{\tau}_{ext}$ includes all other external forces such as gravity. Please refer Tan et al. [124] for details.

6.5 Grasp Controller

The grasp controller brings the hand from its initial state to the state in which the cloth is firmly grasped. The user provides the desired contact points on the cloth and two hand configurations: one for pre-shaping and one for gripping. Using these two input configurations, the algorithm solves inverse kinematics problems to generate two poses, $\mathbf{q}_p$ and $\mathbf{q}_g$, that reach the desired contact points while maintaining the pre-shaping and gripping configurations respectively as close as possible. During

simulation, the grasp controller computes the appropriate torques to move the hand from its arbitrary initial pose to $\mathbf{q}_g$ through $\mathbf{q}_p$, using SPD tracking scheme described in Section 6.4.3.

The path from the initial pose to $\mathbf{q}_p$ can potentially have collisions. The problem rarely occurs in our experiments because we focus on manipulating a single piece of cloth in an uncluttered environment. In the occasion of collision, additional keyframes of hand are added by the user to avoid undesired collisions. We use Catmull-Rom splines to interpolate hand keyframes to achieve smooth motion. If the collision issue occurs more frequently, an automatic path planning algorithm, such as Rapidly-exploring Random Tree (RRT) [77], can be implemented.

6.6 Evaluation

Our hand model is designed based on an anthropomorphic hand structure with 34 degrees of freedom. The simulator is built upon two open-source projects: ARCSim [98, 97, 103] for cloth simulation and DART [81] for multibody simulation. We simulate the two-way coupling of cloth and hand using the contact model proposed by [12].

The window size of the optimization in Equation 48 is selected empirically. We start with the minimum window size and gradually increase it until the resulting motion is above accuracy threshold. In theory, the linearized model will deteriorate as the window size becomes too large, but in practice the upper bound of the window size is often limited by the computation budget. A rough guideline to window size selection is based on how energetic the desired cloth motion is and the distance between controllable and feature vertices.

We evaluate our method by demonstrating a variety of manipulation tasks for real-world activities. In addition, we quantitatively test the accuracy of cloth motion planning algorithms on a random set of input feature trajectories.

6.6.1 Dexterous Manipulation Tasks

We applied our method to various cloth manipulation scenarios shown in Figure 28. The breakdown of computation time, the complexity of the input mesh, and the parameters of the method are listed in Table 2.

Shaking a handkerchief. This example demonstrates the ability of cloth motion planner to control dynamic secondary-motion of cloth. Because the controlling contact points (upper two corners) and the features (bottom two corners) are far apart, the action of the hand does not have direct impact on the motion of the features. Consequently, a seemingly simple feature trajectory can be very difficult to control using the commanding forces from the hand. We demonstrate that our algorithm is able to control two feature trajectories that move asynchronously. To validate our algorithm on a more dynamic motion, we synthesized a sequence where the bottom corners follow a circle around the hands (Figure 31). This motion requires the commanding force to be large enough to overcome gravity but not too large to outpace the timing specified by the trajectories. Finally, we demonstrate that our algorithm is able to handle unexpected external forces robustly as one of the advantages of model-predictive-control. In this example, we added a wind force to the simulation, causing the commanding forces from the hands to compensate.

Wringing a towel. Wringing a towel presents more complex contacts between the fingers/palm and the cloth. Nevertheless, the feature specification can be two simple circular trajectories moving in the opposite directions in the middle part of the towel (Figure 32). Because grasping on a bunched up cloth configuration results in a large number of contact points scattered in a wide range on the cloth, the linear transformation and rigidity rectification are crucial to ensure that the contact points are achievable by the hand.

Folding a T-shirt. Folding a T-shirt in our example requires three contact phases (Figure 29). We show that tasks with regrasping can be achieved by executing the

same algorithm multiple times. Comparing to shaking a handkerchief, folding a T-shirt is an easier example because the features are close in distance to the controlling contact points. For this case, we used small time window for optimization ($K = 3$), and our algorithm can track the feature trajectories very accurately (mean error is 0.003 m).

Putting on a scarf. Putting a scarf on a mannequin causes additional collision forces on the cloth. Because our algorithm considers all other external forces in cloth motion planning (Equation 43), the hands are able to overcome the collision and track the desired trajectories closely (Figure 33).

Lifting from a flat surface. A common strategy to pick up a thin layer of cloth lying flat on a surface is to first use fingers to generate opposing friction forces to create a bulge on the cloth, and grab the bulge to lift the cloth. The feature trajectories include two points on the cloth moving towards middle and four points in the middle of the cloth moving upward (Figure 34). To achieve this task, the thumb and the index finger make two disjoint contact areas with the cloth and move asynchronously towards each other. This example demonstrates that every finger can have its own contact area with the cloth moving in a different direction from other fingers.

6.6.2 Evaluation of Cloth Motion Planning

To validate our algorithm of cloth motion planning, we test our algorithm on a set of randomly sampled feature trajectories. Each trajectory is represented as a b-spline with 22 control points. The space for each trajectory is parameterized by the position and the velocity of the b-spline control points. In our experiments, we define two feature trajectories on the lower corners of a handkerchief while using two upper corners as contact points to provide control. Starting from two asynchronous straight lines as the "mean trajectories", we draw samples from a normal distribution with

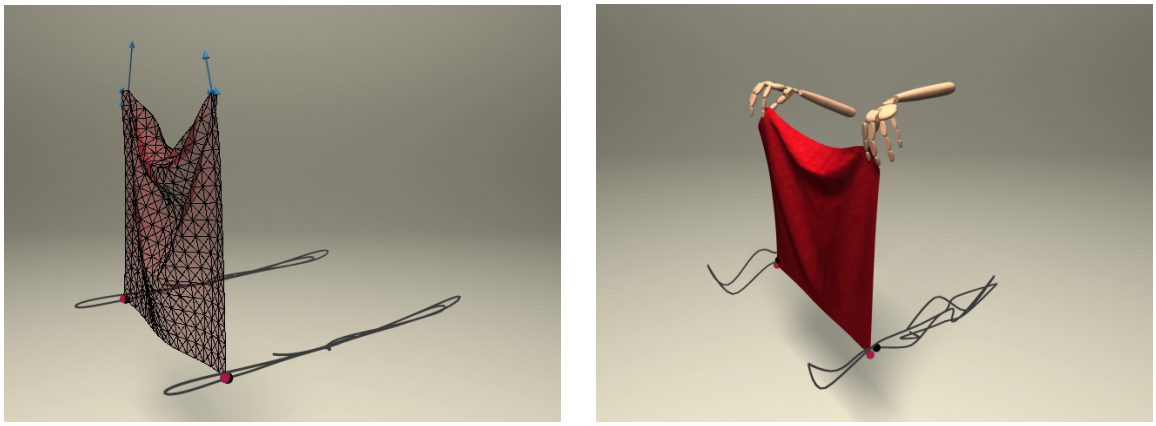
Table 2: Parameters and performance of the examples. #Triangle: the number of triangles in the input mesh. #Vertex: the number of vertices in the input mesh. Window size: window size K used in the optimization. Animation time: wall clock time of the animation. Optimization time: total time for formulating and solving the optimization in cloth motion planning. Control time: total time for solving hand control. Simulation time: total time for cloth and multibody simulation.

Example	#Triangle	#Vertex	Window size	Animation time	Optimization time	Control time	Simulation time
Shaking a handkerchief (asynchronous trajectories)	8192	4225	20	5s	1h56m	37s	21m
Shaking a handkerchief (circle trajectories)	8192	4225	20	5s	3h2m	40s	33m
Shaking a handkerchief (with wind force)	8192	4225	10	6s	27m	45s	24m
Wringing a towel	4078	2071	10	4s	1h18m	1m2s	18m
Folding a T-shirt	26252	13256	3	13s	1h	1m40s	5h14m
Putting on a scarf	11831	6077	3	3.5s	9m	25s	18m
Lifting from a flat surface	8525	4392	10	1.3s	9m	6s	5m

different standard deviations (SD) in the space of b-spline control points.

Because the evaluation only focuses on cloth motion planning and requires a large amount of testing sequences, we make two simplifications in the experiments. First, we apply the optimal contact force $\mathbf{f}_c$ solved from Equation 48 directly to update the state of the cloth, excluding the involvement of hands in this evaluation. Second, we use a piece of cloth with lower resolution to speed up the simulation time. However, the challenge of controlling cloth remains because the number of uncontrolled degrees of freedom is still much more than the controlled degrees of freedom (shown as blue arrows in Figure 35 (a)). Furthermore, we choose an example where the feature points are very far from controlling contact points.

To evaluate the accuracy more precisely, we measure the Euclidean distance between the actual position and the desired position of a feature point. Figure 35 shows the comparison of positions in x-axis for one testing sequence. For an input feature trajectory that has a known control solution (that is the feature trajectory is generated by physical simulation), our motion planning algorithm can achieve mean error less than 0.01 m (Figure 36 (a)), the relative error with respect to the total length of the input trajectory is 0.3%. For randomly sampled feature trajectories, the mean



(a) (b)

Figure 36: (a) Cloth motion optimization evaluation with an input feature trajectory generated by physical simulation. (b) Complete motion evaluation including hand control.

error of the distance is 0.016 m, the relative error with respect to the total length of the input trajectory is 0.8%. The dimension of the cloth is a 0.5 m by 0.5 m square. For completeness, we also apply the entire algorithm including hand control on a selected subset of testing sequences (Figure 36 (b)).

Our cloth motion planning algorithm is independent of the choice of simulators (for example ARCSim). To validate the generality of our algorithm, we tested it on a volumetric mesh simulated as a mass-spring system. Figure 37 shows a rope with a ball hung at the bottom to track a desired trajectory provided by the user. The desired commanding force is solved for the top of the rope (green dot) with our cloth motion planning algorithm, so that the actual motion trajectory of the ball (red curve) can follow the desired trajectory (black curve).

6.7 Limitations

The decision on decoupling the hand and the cloth in the optimization is necessary for performance efficiency, but it also introduces some limitations. In particular, the kinematic constraints of the hand can be approximated in different ways. It could

Figure 37: Our cloth motion planning algorithm is applied on a rope to control the actual trajectory of the ball at the bottom (red curve) to track a desired trajectory drawn by the user (black curve). The commanding force is applied at the top of the rope (green dot).

be more conservative than our current method by limiting the whole hand to move as one rigid body. On the other hand, it could be more aggressive by assuming each finger segment can move independently. We choose a compromised approximation by which each finger can move independently. This approximation is reasonable for the examples demonstrated in this chapter, but it can occasionally result in cloth motion plans unachievable by hands (for example two fingers need to move too far apart).

Our controller is conservative in that it tries to maintain static contact and not exploit slipping or rolling manipulation strategies. We show in our result that this control strategy is sufficient to achieve a wide range of cloth manipulation tasks in daily life. However, for more general cloth manipulation tasks, such as using hands to slide across the cloth to remove wrinkles, slipping manipulation strategy is important.

We take a statistic approach to evaluate the accuracy of cloth motion planning

because analytically defining the successful range of feature trajectories is very difficult if at all possible. This statistical evaluation gives us some confidence that our algorithm is able to produce plausible results for a wide range of input feature trajectories. However, we still can neither guarantee nor predict whether an arbitrary feature trajectory will result in successful dexterous manipulation.

6.8 Conclusion

In this chapter, we introduce a method to enable dexterous manipulation of cloth in physics-based computer animation. Our technique provides a solution to consolidate the control of hand manipulation and the control of cloth simulation. We formulate an optimization problem that solves the commanding forces with respect to the desired cloth motion described by the user and the constraints of the hands. The evaluation shows that our cloth motion planning can accurately achieve the user-specifications on the cloth and can be successfully executed by the simulated hands. We also provide visual demonstration of our technique on a set of cloth manipulation tasks including folding laundry, wringing a towel, and putting on a scarf. We envision that this technique can be integrated with physically simulated full-body characters.

CHAPTER VII

CONCLUSION

In this thesis, we have presented algorithms for synthesizing full-body virtual characters and dexterous hands performing object manipulation tasks. By solving the planning and control problems for virtual characters with a focus on utilizing different body parts and tasks parallelization, our first algorithm creates full-body animation involving concurrent object manipulation tasks. By developing dynamic controllers for both palm and fingers, our second algorithm allows a virtual dexterous hand to reorient objects with different geometries and physical properties. By coupling the simulation of hands and cloth, and solving the control of cloth utilizing the commanding forces from hands, our third algorithm achieves object manipulation which adapts to non-rigid objects. To develop a virtual character with the versatility that humans exhibit in object manipulation, our algorithms focus on the three aspects of versatility that we discuss in Chapter 1.

We introduce a physics-based technique to synthesize human activities involving concurrent full-body manipulation of multiple objects at various locations in a constrained environment. Given an environment map along with the information about the objects and features in the environment, and manipulation graphs that describe all possible strategies to hold, move, push, or release an object, our algorithm generates a continuous animation of a character manipulating multiple objects and environment features concurrently at various locations in a constrained environment. This method enables a virtual character to organize tasks into consecutive and concurrent ones as well as utilize multiple body parts.

Realistic multitasking behaviors can be achieved by better prehension control

algorithms. For example, a person can hook-grasp multiple coffee mugs and open a cabinet door all with one hand. Therefore, imitating humans' versatile manipulation skills by a virtual character requires precise control of dexterous hands. To improve a character's ability of using different body parts within a hand, we develop a technique to manipulate the orientation of an object using both palm and fingers of a virtual hand. The experiments with the virtual Shadow Dexterous Hand model show that the hand is able to pick up a given object on the table, to drop it on a specific spot on the palm, and to let it roll continuously and controllably on the palm, subject to the gravitational and contact forces.

A large repertoire of dexterous hand manipulation tasks involve non-rigid objects, which have not been explored in computer graphics research. To enable a character with the ability of manipulating different object materials especially cloth, we introduce a simulation technique that couples cloth and rigid body simulations to synthesize dexterous manipulation of cloth. This work is intended to be a simple and practical solution to couple existing rigid body simulators and cloth simulators for dexterous manipulation of cloth. On top of this simulation technique, we also introduce a method to consolidate the control of dexterous hand manipulation and the control of cloth simulation. The results show that our technique can be used to create a set of common cloth manipulation tasks including folding laundry, wringing a towel, and putting on a scarf.

As a result, **the algorithms presented in this thesis make a step further towards automatically creating animations for full-body and dexterous hand object manipulation with human versatility.**

7.1 Applications

7.1.1 Animation

In game and film industry, it's typical to generate a character's motion kinematically, by keyframing or blending motions from motion database. Dynamically simulated motion can be more physically-realistic, adapt to different scenarios, and react to user interactions. In this thesis, we present several algorithms to dynamically control and simulate motions of virtual characters in the context of object manipulation. These techniques can be used to generate realistic animations for object manipulation in games and films. Moreover, our algorithms can be applied to create more general motions than object manipulation, where a virtual character needs to interact with the simulated environment and achieve the desired motion. For example, our operational space controller can be used to control a virtual character to achieve tasks in sports video games, such as approaching and tackling a player in the football video game.

Computer animation demands fidelity for the final motion besides a successful control algorithm to achieve the manipulation goal. The fidelity is especially intrinsic for dexterous hand manipulation due to the complexity of the hand motion and the subtle effect during the interaction between a hand and an object. The common practice in the film industry is to manually animate the hand, which usually takes enormous effort, and depends highly on the animators' skills. Researchers have proposed various methods to create realistic motions for dexterous hand manipulation by adapting captured motion to different scenarios, or by synthesizing hand motion with planning and dynamic control algorithms. Even though realistic and detailed animations of dexterous manipulation have been shown in research community in recent years, the objects are typically assumed rigid. This thesis presents algorithms to dynamically synthesize motions of dexterous hands manipulating non-rigid objects to achieve the desired motion target.

7.1.2 Virtual Reality

The physics-based simulation of interactions between hands and objects can be used in virtual reality (VR) applications such as haptics research, training for manual assembly, and game applications [56, 86, 106]. Thanks to the current development of VR headsets (for example Oculus Rift) with the ability of hand tracking, people can interact with 3D objects in the virtual environment [27]. By utilizing VR devices, the control of virtual hands can be more direct and intuitive for user to interact with virtual objects compared with traditional UI devices such as mouse and keyboard. Our algorithms for synthesizing motions of cloth manipulation can be used in VR applications to enable non-rigid object manipulation.

7.1.3 Robotics

Problems involving object manipulation have been frequently addressed in robotics research. One challenge in robotics is that the control algorithms are often constrained by the hardware design. As a result, existing algorithms for both full-body and dexterous hand manipulation only utilize designated parts of robots for object manipulation. In contrast, humans are much more versatile by using different body parts, such as carrying an object on the shoulder, on the back, or even on the head. As a result, tasks can be achieved in parallel for humans, which makes human motion very efficient for object manipulation tasks. Our algorithms enable virtual characters with the level of versatility that humans exhibit by developing control algorithms for both full-body concurrent manipulation tasks and dexterous hand manipulation tasks. The algorithms can potentially be used on a real robot to make it more efficient and more humanlike.

7.2 Limitations

In Chapter 3, we assume that the manipulation tasks are primarily done by the upper body and locomotion is done by the lower body. However, for task such as lifting a heavy object, coordination between locomotion and upper body manipulation is vital. In addition, our algorithm only considers the shortest distance when planning the events. Other additional criteria, such as the amount of effort required for each task, could be taken into account.

In Chapter 4, one limitation of our approach is that the object cannot be too different from a prism. For example, rolling a key on the palm would be a challenging case. Also, the detailed information about the object must be known in advance.

In Chapter 5, we make the decision not to implement accurate two-way coupling for the reason of having a simple implementation and efficient computation. However, we do recognize that some examples like putting on a sock requires that the rigid bodies react to the dynamics of the cloth accurately.

In Chapter 6, our controller is conservative in that it tries to maintain static contact and not exploit slipping or rolling manipulation strategies. This choice is based on the observation that most of the common cloth manipulation tasks are achieved through static contact between hands and cloth. However, for more general cloth manipulation tasks, such as using hands to slide across the cloth to remove wrinkles, the slipping manipulation strategy is important.

The algorithms presented in this thesis are only evaluated on simplified human models represented as articulated rigid body systems. Researchers have developed more accurate human models with detailed simulation of tendons and muscles [120, 131] and demonstrated their impact on control of manipulation tasks [111]. Jain and Liu [55] have presented a technique to combine a deformable skin with articulated rigid body systems. The traditional articulated rigid body models appear to be a perfect balance between fidelity and complexity. However, the anatomical detail

of the model can become essential when it plays an important role in determining the naturalness of motions, for example, when muscle synergies affect the realism of motions. Moreover, the deformation of the body parts can also be necessary when it has an impact on the success of manipulation tasks, for example, when the task can only be achieved by a large contact area generated by the deformation of a body.

7.3 Future Work

7.3.1 Short Term

This thesis presents several algorithms for synthesizing human motions for object manipulation. The algorithms focus on either the upper body motion with arms (Chapter 3), or hand motion with fingers (Chapter 4, 5, and 6). One future direction would be combining our algorithms on a fully simulated full-body character with dexterous hands. As a study by Joerg et al. [58] shows, even very subtle desynchronization errors in body-and-hand motions can be detected by the human eye. To be able to create motions that are more complex and adaptive to more general object manipulation tasks, the challenging problem of coordination between the full-body motion and the dexterous hand motion needs to be solved.

Another interesting future direction is to integrate our manipulation controller with other full-body motion controllers. We are particularly interested in integrating our upper body controller (Chapter 3) with the biped walking controller developed by Coros et al. [30]. Their work has demonstrated a variety of walking related skills, such as picking up and carrying objects. Physically simulating whole-body manipulation can raise new challenges in balance control. In addition, another exciting future direction is to incorporate our algorithms on dexterous hand manipulation of cloth (Chapter 5 and 6) with the recent work on full-body character controller for animating character dressing [29]. Such an augmented system would allow more cloth manipulation tasks which require full-body motion. Developing a virtual character

with dexterous hands capable of folding laundry and dressing can be a fruitful future research direction.

7.3.2 Long Term

The algorithms presented in this thesis are evaluated on virtual characters in the simulated environment. With more research on making the simulated world consistent to the real world, connecting the computer animation and robotics areas will be more promising. As a result, our work has potential to be applied on real physical robots in order to make robots' motions more versatile in object manipulation tasks and more efficient when assisting people.

REFERENCES

[1] "ARCSim, http://graphics.berkeley.edu/resources/arcsim/."

[2] "Bullet Physics Library, http://bulletphysics.org/wordpress/."

[3] "DART: Dynamic Animation and Robotics Toolkit", https://github.com/dartsim/dart/."

[4] "Gazebo,http://gazebosim.org/."

[5] "Maya, http://www.autodesk.com/products/autodesk-maya/overview."

[6] "Open Dynamics Engine, http://www.ode.org."

[7] Abe, Y. and Popović, J., "Interactive animation of dynamic manipulation," in *Eurographics/SIGGRAPH Symposium on Computer Animation*, 2006.

[8] Abell, T. and Erdmann, M. A., "Stably supported rotations of a planar polygonwith two frictionless contacts," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995.

[9] Ainsley, S., Vouga, E., Grinspun, E., and Tamstorf, R., "Speculative parallel asynchronous contact mechanics," *ACM Trans. Graph. (SIGGRAPH Asia 2012)*, vol. 31, no. 6, p. 151, 2012.

[10] Aiyama, Y., Inaba, M., and Inoue, H., "Pivoting: A new method of graspless manipulation of object by robot fingers," in *Proc. 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 00, pp. 136–143, 1993.

[11] Andrews, S. and Kry, P. G., "Goal directed multi-finger manipulation: Control policies and analysis," *Computers & Graphics*, vol. 37, no. 7, pp. 830–839, 2013.

[12] Bai, Y. and Liu, C. K., "Coupling cloth and rigid bodies for dexterous manipulation," in *Proceedings of the Seventh International Conference on Motion in Games*, pp. 139–145, ACM, 2014.

[13] Bai, Y. and Liu, C., "Dexterous manipulation using both palm and fingers," in *Proc. 2014 IEEE International Conference on Robotics and Automation*, IEEE, 2014.

[14] Bai, Y., Siu, K., and Liu, C. K., "Synthesis of concurrent object manipulation tasks," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 156, 2012.

[15] Baraff, D. and Witkin, A., "Large steps in cloth simulation," in *Proc. SIGGRAPH 98*, pp. 43–54, ACM, 1998.

[16] Baraff, D., Witkin, A., and Kass, M., "Untangling cloth," *ACM Trans. Graph. (SIGGRAPH 2003)*, vol. 22, no. 3, pp. 862–870, 2003.

[17] Barbič, J., Sin, F., and Grinspun, E., "Interactive editing of deformable simulations," in *ACM Trans. Graph. (SIGGRAPH 2012)*, vol. 31, p. 70, ACM, 2012.

[18] Bergou, M., Mathur, S., Wardetzky, M., and Grinspun, E., "Tracks: toward directable thin shells," in *ACM Trans. Graph. (SIGGRAPH 2007)*, vol. 26, p. 50, ACM, 2007.

[19] Bersch, C., Pitzer, B., and Kammel, S., "Bimanual robotic cloth manipulation for laundry folding," in *Proc. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1413–1419, IEEE, 2011.

[20] Bicchi, A. and Sorrentino, R., "Dexterous manipulation through rolling," in *Proc. 1995 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 452–457, Ieee, 1995.

[21] Bicchi, A., "Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 652–662, 2000.

[22] Boutselis, G. I., Bechlioulis, C. P., Liarokapis, M. V., and Kyriakopoulos, K. J., "An integrated approach towards robust grasping with tactile sensing," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 3682–3687, IEEE, 2014.

[23] Bridson, R., Fedkiw, R., and Anderson, J., "Robust treatment of collisions, contact and friction for cloth animation," *ACM Trans. Graph. (SIGGRAPH 2002)*, vol. 21, no. 3, pp. 594–603, 2002.

[24] Bridson, R., Marino, S., and Fedkiw, R., "Simulation of clothing with folds and wrinkles," in *Proc. 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 28–36, Eurographics Association, 2003.

[25] Brock, D., "Enhancing the dexterity of a robot hand using controlled slip," in *Proc. 1988 IEEE International Conference on Robotics and Automation*, no. 7, pp. 249–251, IEEE Comput. Soc. Press, 1988.

[26] Bruderlin, A. and Williams, L., "Motion signal processing," in *SIGGRAPH*, pp. 97–104, Aug. 1995.

[27] Chagué, S., "Real virtuality: immersive explorers," in *ACM SIGGRAPH 2015 Computer Animation Festival*, pp. 195–195, ACM, 2015.

[28] Choi, M. G., Lee, J., and Shin, S. Y., "Planning biped locomotion using motion capture data and probabilistic roadmaps," *ACM Trans. Graph.*, vol. 22, pp. 182–203, April 2003.

[29] Clegg, A., Tan, J., Turk, G., and Liu, C. K., "Animating human dressing," in *ACM Trans. Graph. (SIGGRAPH 2015)*, vol. 34, ACM, 2015.

[30] Coros, S., Beaudoin, P., and van de Panne, M., "Generalized biped walking control," in *ACM Transactions on Graphics (TOG)*, vol. 29, p. 130, ACM, 2010.

[31] Cusumano-Towner, M., Singh, A., Miller, S., O'Brien, J. F., and Abbeel, P., "Bringing clothing into desired configurations with limited perception," in *Proc. 2011 IEEE International Conference on Robotics and Automation*, pp. 3893–3900, IEEE, 2011.

[32] Dafle, N. C., Rodriguez, A., Paolini, R., Tang, B., Srinivasa, S. S., Erdmann, M., Mason, M. T., Lundberg, I., Staab, H., and Fuhlbrigge, T., "Regrasping objects using extrinsic dexterity," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 2560–2560, IEEE, 2014.

[33] de Lasa, M. and Hertzmann, A., "Prioritized optimization for task-space control," in *IEEE/RSJ international conference on Intelligent robots and systems*, IROS'09, pp. 5755–5762, 2009.

[34] de Lasa, M., Mordatch, I., and Hertzmann, A., "Feature-based locomotion controllers," *ACM Trans. Graph.*, vol. 29, 2010.

[35] Dogar, M. and Srinivasa, S., "Push-grasping with dexterous hands: Mechanics and a method," in *Proc. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2010.

[36] Erdmann, M. A., "An exploration of nonprehensile two-palm manipulation: Planning and execution," *International Journal of Robotics Research*, 1995.

[37] Erdmann, M. A., Mason, M. T., and Vaněček, G. in *Mechanical Parts Orienting : The Case of a Polyhedron on a Table*, pp. 360–365, Ieee, 1991.

[38] Erdmann, M. and Mason, M., "An exploration of sensorless manipulation," *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, pp. 369–379, 1988.

[39] Fahantidis, N., Paraschidis, K., Petridis, V., Doulgeri, Z., Petrou, L., and Hasapis, G., "Robot handling of flat textile materials," *Robotics & Automation Magazine, IEEE*, vol. 4, no. 1, pp. 34–41, 1997.

[40] Feng, A. W., Xu, Y., and Shapiro, A., "An example-based motion synthesis technique for locomotion and object manipulation," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 95–102, ACM, 2012.

[41] Gill, P., Saunders, M., and Murray, W., "Snopt: An sqp algorithm for large-scale constrained optimization," Tech. Rep. NA 96-2, University of California, San Diego, 1996.

[42] Grinspun, E., Hirani, A. N., Desbrun, M., and Schröder, P., "Discrete shells," in *Proc. 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 62–67, Eurographics Association, 2003.

[43] Ha, S., Bai, Y., and Liu, C. K., "Human motion reconstruction from force sensors," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 129–138, ACM.

[44] Han, L., Guan, Y., and Li, Z., "Dextrous manipulation with rolling contacts," in *Proc. 1997 IEEE International Conference on Robotics and Automation*, pp. 2–7, 1997.

[45] Han, L. and Trinkle, J., "Dextrous manipulation by rolling and finger gaiting," in *Proc. 1998 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 730–735, Ieee, 1998.

[46] Harada, K., Kajita, S., Kaneko, K., and Hirukawa, H., "Pushing manipulation by humanoid considering two-kinds of zmps," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2, pp. 1627–1632, IEEE, 2003.

[47] Harmon, D., Vouga, E., Smith, B., Tamstorf, R., and Grinspun, E., "Asynchronous contact mechanics," in *ACM Trans. Graph. (SIGGRAPH 2009)*, vol. 28, p. 87, ACM, 2009.

[48] Harmon, D., Vouga, E., Tamstorf, R., and Grinspun, E., "Robust treatment of simultaneous collisions," *ACM Trans. Graph. (SIGGRAPH 2008)*, vol. 27, no. 3, p. 23, 2008.

[49] Hildebrandt, K., Schulz, C., von Tycowicz, C., and Polthier, K., "Interactive spacetime control of deformable objects," *ACM Trans. Graph. (SIGGRAPH 2012)*, vol. 31, no. 4, p. 71, 2012.

[50] Ho, E. S. and Komura, T., "Character motion synthesis by topology coordinates," in *Computer Graphics Forum*, vol. 28, pp. 299–308, Wiley Online Library, 2009.

[51] Horn, B. K., "Closed-form solution of absolute orientation using unit quaternions," *JOSA A*, vol. 4, no. 4, pp. 629–642, 1987.

[52] Huang, W. and Mason, M. T., "Experiments in impulsive manipulation," in *Proc. 1998 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1077 – 1082, May 1998.

[53] Huang, W. and Mason, M. T., "Mechanics, planning, and control for tapping," *International Journal of Robotics Research*, vol. 19, pp. 883–894, October 2000.

[54] Huang, Y., Mahmudi, M., and Kallmann, M., "Planning humanlike actions in blending spaces," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 2653–2659, IEEE, 2011.

[55] Jain, S. and Liu, C. K., "Controlling physics-based characters using soft contacts," in *ACM Transactions on Graphics (TOG)*, vol. 30, p. 163, ACM, 2011.

[56] James, D. L. and Pai, D. K., "A unified treatment of elastostatic contact simulation for real time haptics," *Haptics-e, The Electronic Journal of Haptics Research*, vol. 2, no. 1, 2001.

[57] Jansson, J. and Vergeest, J. S., "Combining deformable-and rigid-body mechanics simulation," *The Visual Computer*, vol. 19, no. 5, pp. 280–290, 2003.

[58] Jörg, S., Hodgins, J., and O'Sullivan, C., "The perception of finger motions," in *Proceedings of the 7th Symposium on Applied Perception in Graphics and Visualization*, pp. 129–133, ACM, 2010.

[59] Kaldor, J. M., James, D. L., and Marschner, S., "Efficient yarn-based cloth with adaptive contact linearization," *ACM Trans. Graph. (SIGGRAPH 2010)*, vol. 29, no. 4, p. 105, 2010.

[60] Kallmann, M., "Interaction with 3-d objects," in *Handbook of Virtual Humans*, pp. 303–322, 2004.

[61] Kallmann, M., "Scalable solutions for interactive virtual humans that can manipulate objects," in *AIIDE*, pp. 69–75, 2005.

[62] Kallmann, M., Aubel, A., Abaci, T., and Thalmann, D., "Planning collision-free reaching motions for interactive object manipulation and grasping," in *Computer Graphics Forum*, vol. 22, pp. 313–322, Wiley Online Library, 2003.

[63] Kaufman, D. M., Sueda, S., James, D. L., and Pai, D. K., "Staggered projections for frictional contact in multibody systems," in *ACM Trans. Graph. (SIGGRAPH Asia 2008)*, vol. 27, p. 164, ACM, 2008.

[64] Kavraki, L., Svestka, P., Latombe, J.-C., and Overmars, M. H., "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[65] Kent, B. A., Kakish, Z. M., Karnati, N., and Engeberg, E. D., "Adaptive synergy control of a dexterous artificial hand to rotate objects in multiple orientations via emg facial recognition," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 6719–6725, IEEE, 2014.

[66] Khatib, O., "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 43 –53, february 1987.

[67] Khatib, O., Sentis, L., Park, J., and Warren, J., "Whole-body dynamic behavior and control of human-like robots," *International Journal of Humanoid Robotics*, vol. 1, no. 1, pp. 29–43, 2004.

[68] Kim, Y.-J., Lee, Y., Kim, J., Lee, J.-W., Park, K.-M., Roh, K.-S., and Choi, J.-Y., "Roboray hand: A highly backdrivable robotic hand with sensorless contact force measurements," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 6712–6718, IEEE, 2014.

[69] Koga, Y., Kondo, K., Kuffner, J., and Latombe, J.-C., "Planning motions with intentions," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 395–408, ACM, 1994.

[70] Kopicki, M., Detry, R., Schmidt, F., Borst, C., Stolkin, R., and Wyatt, J. L., "Learning dexterous grasps that generalise to novel objects by combining hand and contact models," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 5358–5365, IEEE, 2014.

[71] Kosuge, K., Yoshida, H., Fukuda, T., Sakai, M., and Kanitani, K., "Manipulation of a flexible object by dual manipulators," in *Proc. 1995 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 318–323, IEEE, 1995.

[72] Kovar, L., Gleicher, M., and Pighin, F., "Motion graphs," in *SIGGRAPH*, pp. 473–482, July 2002.

[73] Kry, P. G. and Pai, D. K., "Interaction capture and synthesis," in *ACM Trans. Graph. (SIGGRAPH 2006)*, vol. 25, pp. 872–880, ACM, 2006.

[74] Lamarche, F. and Donikian, S., "The orchestration of behaviours using resources and priority levels," in *Eurographic workshop on Computer animation and simulation*, pp. 171–182, 2001.

[75] Lau, M. and Kuffner, J. J., "Behavior planning for character animation," in *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 271–280, 2005.

[76] Lavalle, S. M. and Kuffner, J. J., "Rapidly-exploring random trees: Progress and prospects," Aug. 2000.

[77] LaValle, S. M. and Kuffner, J. J., "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[78] Li, M., Yin, H., Tahara, K., and Billard, A., "Learning object-level impedance control for robust grasping and dexterous manipulation," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 6784–6791, IEEE, 2014.

[79] Liu, C. K., "Synthesis of interactive hand manipulation," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 163–171, Eurographics Association, 2008.

[80] Liu, C. K., "Dextrous manipulation from a grasping pose," in *ACM Trans. Graph. (SIGGRAPH 2009)*, vol. 28, 2009.

[81] Liu, C. K. and Jain, S., "A short tutorial on multibody dynamics," Tech. Rep. GIT-GVU-15-01-1, Georgia Institute of Technology, School of Interactive Computing, 08 2012.

[82] Liu, T., Bargteil, A. W., O'Brien, J. F., and Kavan, L., "Fast simulation of mass-spring systems," *ACM Trans. Graph. (SIGGRAPH Asia 2013)*, vol. 32, no. 6, p. 214, 2013.

[83] Liu, Y. and Badler, N. I., "Real-time reach planning for animated characters using hardware acceleration," in *Computer Animation and Social Agents, 2003. 16th International Conference on*, pp. 86–93, IEEE, 2003.

[84] Lynch, K. M. and Mason, M. T., "Dynamic nonprehensile underactuated manipulation," in *Proc. 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[85] Ma, R. R. and Dollar, A. M., "On dexterity and dexterous manipulation," in *Proc. 2011 IEEE International Conference on Robotics and Automation*, pp. 1–7, Ieee, June 2011.

[86] Magnenat-Thalmann, N., Volino, P., Bonanni, U., Summers, I. R., Bergamasco, M., Salsedo, F., and Wolter, F.-E., "From physics-based simulation to the touching of textiles: The haptex project," *IJVR*, vol. 6, no. 3, pp. 35–44, 2007.

[87] Mason, M. T., "Progress in nonprehensile manipulation," *The International Journal of Robotics Research*, 1999.

[88] Miguel, E., Feng, A., Xu, Y., Shapiro, A., Tamstorf, R., Bradley, D., Schvartzman, S. C., Thomaszewsky, B., Bickel, B., Matusik, W., and others, "Towards cloth-manipulating characters," in *Proc. of CEIG (Spanish Computer Graphics Conference)*, 2010.

[89] Miguel, E. and Otaduy, M. A., "Efficient simulation of contact between rigid and deformable objects," *Multibody Dynamics*, 2011.

[90] Miller, S., Van Den Berg, J., Fritz, M., Darrell, T., Goldberg, K., and Abbeel, P., "A geometric approach to robotic laundry folding," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 249–267, 2012.

[91] Mistry, M. and Righetti, L., "Operational space control of constrained and underactuated systems," in *Proceedings of Robotics: Science and Systems*, (Los Angeles, CA, USA), June 2011.

[92] Mitsui, K. and Ozawa, R., "Design of a tendon-driven robotic hand with an embedded camera," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 6733–6738, IEEE, 2014.

[93] Mordatch, I., Popović, Z., and Todorov, E., "Contact-invariant optimization for hand manipulation," in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pp. 137–144, Eurographics Association, 2012.

[94] Müller, M. and Gross, M., "Interactive virtual materials," in *Proc. Graphics Interface 2004*, pp. 239–246, Canadian Human-Computer Communications Society, 2004.

[95] Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J., "Position based dynamics," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 109–118, 2007.

[96] Nakamura, Y., Hanafusa, H., and Yoshikawa, T., "Task-priority based redundancy control of robot manipulators," *Int. J. Rob. Res.*, vol. 6, pp. 3–15, July 1987.

[97] Narain, R., Pfaff, T., and O'Brien, J. F., "Folding and crumpling adaptive sheets," *ACM Trans. Graph. (SIGGRAPH 2013)*, vol. 32, no. 4, p. 51, 2013.

[98] Narain, R., Samii, A., and O'Brien, J. F., "Adaptive anisotropic remeshing for cloth simulation," *ACM Trans. Graph. (SIGGRAPH Asia 2012)*, vol. 31, no. 6, p. 152, 2012.

[99] Nishiwaki, K., Yoon, W.-K., and Kagami, S., "Motion control system that realizes physical interaction between robot's hands and environment during walk," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pp. 542–547, IEEE, 2006.

[100] Osawa, F., Seki, H., and Kamiya, Y., "Unfolding of massive laundry and classification types by dual manipulator," *JACIII*, vol. 11, no. 5, pp. 457–463, 2007.

[101] Otaduy, M. A., Tamstorf, R., Steinemann, D., and Gross, M., "Implicit contact handling for deformable objects," in *Computer Graphics Forum*, vol. 28, pp. 559–568, Wiley Online Library, 2009.

[102] Pekarovskiy, A., Saluja, K., Sarkar, R., and Buss, M., "Resonance-driven dynamic manipulation: Dribbling and juggling with elastic beam," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 943–948, IEEE, 2014.

[103] Pfaff, T., Narain, R., de Joya, J. M., and O'Brien, J. F., "Adaptive tearing and cracking of thin sheets," in *ACM Trans. Graph. (SIGGRAPH 2014)*, vol. 33, p. 110, ACM, 2014.

[104] Pokorny, F. T., Bekiroglu, Y., and Kragic, D., "Grasp moduli spaces and spherical harmonics," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 389–396, IEEE, 2014.

[105] Pollard, N. S. and Zordan, V. B., "Physically based grasping control from example," in *Proc. 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 311–318, ACM, 2005.

[106] Prachyabrued, M. and Borst, C., "Design and evaluation of visual interpenetration cues in virtual grasping," *Visualization and Computer Graphics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[107] Rijpkema, H. and Girard, M., "Computer animation of knowledge-based human grasping," *SIGGRAPH Comput. Graph.*, vol. 25, pp. 339–348, July 1991.

[108] Roa, M. A., Chen, Z., Staal, I. C., Muirhead, J. N., Maier, A., Pleintinger, B., Borst, C., and Lii, N. Y., "Towards a functional evaluation of manipulation performance in dexterous robotic hand design," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 6800–6807, IEEE, 2014.

[109] Rose, C., Cohen, M. F., and Bodenheimer, B., "Verbs and adverbs: Multidimensional motion interpolation," *IEEE Computer Graphics and Applications*, vol. 18, Sept. 1998.

[110] Rus, D., "Dexterous rotations of polyhedra," in *Proc. 1992 IEEE International Conference on Robotics and Automation*, pp. 2758–2763, IEEE Comput. Soc. Press, 1992.

[111] Sachdeva, P., Sueda, S., Bradley, S., Fain, M., and Pai, D. K., "Biomechanical simulation and control of hands and tendinous systems," in *ACM Trans. Graph. (SIGGRAPH 2015)*, vol. 34, ACM, 2015.

[112] Sawasaki, N., Inaba, M., and Inoue, H., "Tumbling objects using a multi-fingered robot," in *Proc. the 20th International Symposium on Industrial Robots and Robot Exhibition*, pp. 609–616, Ieee, 1987.

[113] Sentis, L. and Khatib, O., "Control of free-floating humanoid robots through task prioritization," in *IEEE International Conference on Robotics and Automation*, 2005.

[114] Shapiro, A., Kallmann, M., and Faloutsos, P., "Interactive motion correction and object manipulation," in *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pp. 137–144, ACM, 2007.

[115] Shinar, T., Schroeder, C., and Fedkiw, R., "Two-way coupling of rigid and deformable bodies," in *Proc. 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 95–103, Eurographics Association, 2008.

[116] Sifakis, E., Shinar, T., Irving, G., and Fedkiw, R., "Hybrid simulation of deformable solids," in *Proc. 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 81–90, Eurographics Association, 2007.

[117] Srinivasa, S., Erdmann, M. A., and Mason, M. T., "Using projected dynamics to plan dynamic contact manipulation," in *Proc. 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[118] Stewart, D. E. and Trinkle, J. C., "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," *International Journal for Numerical Methods in Engineering*, vol. 39, no. 15, pp. 2673–2691, 1996.

[119] Stilman, M., Schamburek, J.-U., Kuffner, J., and Asfour, T., "Manipulation planning among movable obstacles," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 3327–3332, IEEE, 2007.

[120] Sueda, S., Kaufman, A., and Pai, D. K., "Musculotendon simulation for hand animation," in *ACM Trans. Graph. (SIGGRAPH 2008)*, vol. 27, p. 83, ACM, 2008.

[121] Sunada, C., Argaez, D., Dubowsky, S., and Mavroidis, C., "A coordinated jacobian transpose control for mobile multi-limbed robotic systems," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 1910–1915, IEEE, 1994.

[122] Sung, J., Selman, B., and Saxena, A., "Learning sequences of controllers for complex manipulation tasks," in *ICML workshop on Prediction with Sequential Models*, Citeseer, 2013.

[123] Takubo, T., Inoue, K., and Arai, T., "Pushing an object considering the hand reflect forces by humanoid robot in dynamic walking," in *Robotics and*

Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, pp. 1706–1711, IEEE, 2005.

[124] Tan, J., Liu, K., and Turk, G., "Stable proportional-derivative controllers," *Computer Graphics and Applications, IEEE*, vol. 31, no. 4, pp. 34–44, 2011.

[125] Tang, M., Manocha, D., and Tong, R., "Fast continuous collision detection using deforming non-penetration filters," in *Proc. 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pp. 7–13, ACM, 2010.

[126] Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K., "Elastically deformable models," in *Proc. SIGGRAPH 87*, vol. 21, pp. 205–214, ACM, 1987.

[127] Tournassoud, P., Lozano-Perez, T., and Mazer, E., "Regrasping," in *Proc. 1987 IEEE International Conference on Robotics and Automation*, pp. 1924–1928, Ieee, 1987.

[128] Wang, H. and Komura, T., "Manipulation of flexible objects by geodesic control," in *Computer Graphics Forum*, vol. 31, pp. 499–508, Wiley Online Library, 2012.

[129] Wang, H., Sidorov, K. A., Sandilands, P., and Komura, T., "Harmonic parameterization by electrostatics," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 5, p. 155, 2013.

[130] Wang, H., Ramamoorthi, R., and O'Brien, J. F., "Data-driven elastic models for cloth: Modeling and measurement," *ACM Trans. Graph. (SIGGRAPH 2011)*, vol. 30, pp. 71:1–11, July 2011.

[131] Wang, J. M., Hamner, S. R., Delp, S. L., and Koltun, V., "Optimizing locomotion controllers using biologically-based actuators and objectives.," *ACM Trans. Graph.*, vol. 31, no. 4, p. 25, 2012.

[132] Wang, Y., Min, J., Zhang, J., Liu, Y., Xu, F., Dai, Q., and Chai, J., "Video-based hand manipulation capture through composite motion control," in *ACM Trans. Graph. (SIGGRAPH 2013)*, vol. 32, p. 43, ACM, 2013.

[133] Wojtan, C., Mucha, P. J., and Turk, G., "Keyframe control of complex particle systems using the adjoint method," in *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 15–23, Eurographics Association, 2006.

[134] Wu, J., LUO, Z., Yamakita, M., and Ito, K., "Adaptive hybrid control of manipulators on uncertain flexible objects," *Advanced Robotics*, vol. 10, no. 5, pp. 469–485, 1995.

[135] Yamane, K., Kuffner, J. J., and Hodgins, J. K., "Synthesizing animations of human manipulation tasks," in *ACM Transactions on Graphics (TOG)*, vol. 23, pp. 532–539, ACM, 2004.

[136] Ye, Y. and Liu, C. K., "Synthesis of detailed hand manipulations using contact sampling," in *ACM Trans. Graph. (SIGGRAPH 2012)*, vol. 31, pp. 1–10, ACM, 2012.

[137] Yoshida, E., Belousov, I., Esteves, C., and Laumond, J.-P., "Humanoid motion planning for dynamic tasks," in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pp. 1–6, IEEE, 2005.

[138] Zhao, W., Zhang, J., Min, J., and Chai, J., "Robust realtime physics-based motion control for human grasping," in *ACM Trans. Graph. (SIGGRAPH ASIA 2013)*, vol. 32, p. 207, ACM, 2013.

[139] Zumel, N. B. and Erdmann, M. A. in *Nonprehensile two palm manipulation with non-equilibrium transitions between stable states*, pp. 3317–3323, Ieee, 1996.

VITA

Yunfei Bai's research area includes computer graphics, computer animation, physics-based simulation, robotics, and machine learning. Yunfei Bai completed his Bachelor of Engineer degree in Automation from Tsinghua University, Beijing, China, in 2010. In the same year, he joined Computer Graphics Group at Georgia Institute of Technology to pursue his doctoral research titled Simulating Human Motion for Object Manipulation. At Georgia Tech, Yunfei has been involved in several research projects related to creating character animation for object manipulation at Computer Graphics Group advised by Professor C.Karen Liu, producing dynamically consistent walking motion collaborated with Comparative Neuromechanics Laboratory, and developing artist-directed dynamics tool for 2D animation collaborated with Adobe Research.

By the time of graduation, Yunfei's research findings have been published in several journal and conference proceedings [14, 13, 12, 43]. In addition, two journal papers are under review/submission. His paper received "Best Student Paper Award" at ACM SIGGRAPH Conference on Motion in Games (MIG), LA, 2014.